Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/6180--01-8572

Supervisory Control System for Ship Damage Control: Volume 2 — Scenario Generation and Physical Ship Simulation of Fire, Smoke, Flooding, and Rupture

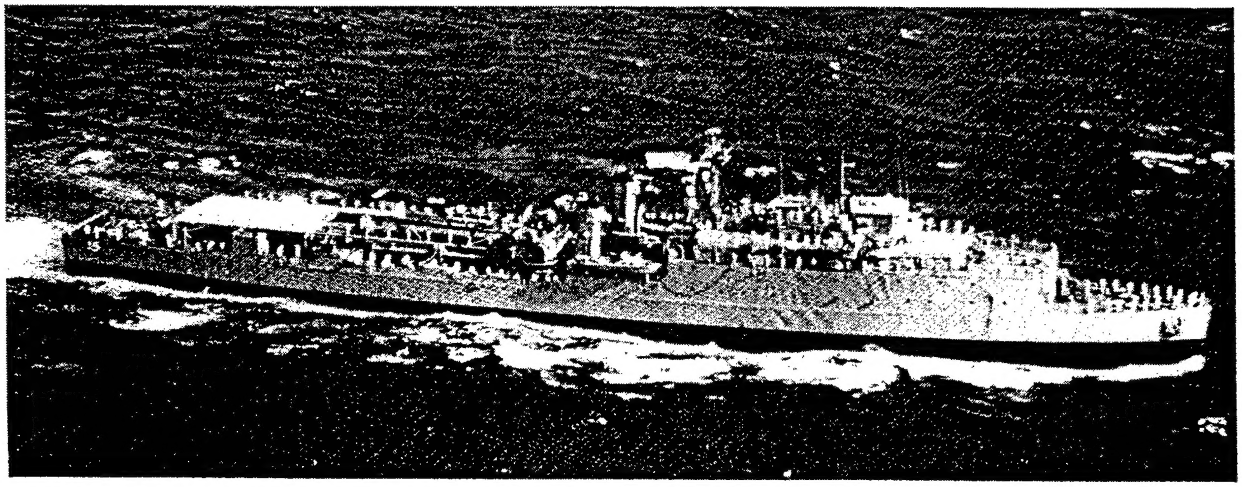
GUOMING SHOU
DAVID C. WILKINS
MARK HOEMMEN
CHRISTOPER MUELLER

Beckman Institute
University of Illinois, Urbana, Illinois

PATRICIA A. TATEM
FREDERICK W. WILLIAMS

Navy Technology Center for Safety and Survivability
Chemistry Division

August 24, 2001



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE August 24, 2001	3. REPORT TYPE AND DATES COVERED Final FY 1999-FY 2001

4. TITLE AND SUBTITLE Supervisory Control System for Ship Damage Control: Volume 2— Scenario Generation and Physical Ship Simulation of Fire, Smoke, Flooding, and Rupture	5. FUNDING NUMBERS PE - 63508N
6. AUTHOR(S) G. Shou,* D.C. Wilkins,* M. Hoemmen,* C. Mueller,* P.A. Tatem, and F.W. Williams	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Research Laboratory, Code 6180 4555 Overlook Avenue, SW Washington, DC 20375-5320	8. PERFORMING ORGANIZATION REPORT NUMBER NRL/MR/6180--01-8572
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5660	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES *Beckman Institute, University of Illinois, Urbana IL 61801	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report describes a computer program for Damage Control Simulation (DC-SIM). This DC-SIM program provides for specification and real-time simulation of physical ship processes associated with ship crises, including fire and smoke spread, hull rupture, fire main pipe rupture, multi-deck progressive flooding, sub-system deactivation and ship stability. Also simulated are various methods of crisis suppression management.

14. SUBJECT TERMS Flooding Fire	Damage control Supervisory control	Automation	15. NUMBER OF PAGES 86 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

Supervisory Control System for Ship Damage Control: Volume 2 – Scenario Generation and Physical Ship Simulation of Fire, Smoke, Flooding and Rupture

1. Introduction

This report describes a computer program for Damage Control Simulation (DC-SIM). The DC-SIM program provides for specification and real-time simulation of physical ship processes associated with ship crises, including fire and smoke spread, hull rupture, fire main pipe rupture, multi-deck progressive flooding, sub system deactivation, and ship stability. Also simulated are various methods of crisis suppression management, including water mist, Aqueous Film-Forming Foam (AFFF), heptafluoro propane (HFP) and desmoking. The primary relevance of DC-SIM is to the domain of ship damage control.

One capacity in which DC-SIM is currently being used is for training of Damage Control Assistants (DCAs) at the Sea Warfare Officer School (SWOS) in Newport, RI. Toward this end, DC-SIM serves as a component of an immersive damage control trainer called DC-TRAIN. DC-SIM allows placing a ship in a crisis state and then simulating the progression of the crisis through time based on the laws of physics and the damage control response actions taken by ship personnel. The solving of multiple scenarios provides the DCA with training that would not otherwise be available, since it is cost-prohibitive to repeatedly inflict major damage to a ship in order to provide DCAs with whole-task training in real-time crisis management.

Another capacity in which the DC-SIM system is currently used is for development of the next generation of automated control algorithms for ship damage control. This effort is part of the Damage Control - Automation for Reduce Manning (DC-ARM) program of the Naval Research Laboratory (NRL), [Parker et.al., 1999]. Toward this end, DC-SIM is being used to assist in the development of a supervisory control system for damage control (DC-SCS), by exercising it during the research and development phase. The DC-SCS is also being tested in a live fire environment aboard the ex-USS *Shadwell* off the coast of Mobile, Alabama. DC-SIM complements the exercising of DC-SCS aboard the ex-USS *Shadwell* [Carhart, et.al., 1992] by permitting simulation of more complex disasters and a far greater number of disasters than is possible in a live test facility. It is also planned to use DC-SIM to provide a means of training a DCA in the use of the DC-SCS aboard the ex-USS *Shadwell*; repeated use of the DC-SCS increases the extent that the DCA can acquire a situation assessment of the ship crisis through the DC-SCS, and also allow the DCA to know when and how to override the DC-SCS when its behavior appears incorrect or inappropriate for the situation.

DC-SIM is able to simulate a wide-range of crisis scenarios. Peacetime scenarios are important because this is when most ship damage crises currently occur. Wartime crises are important because effective damage control facilitates allowing the ship to achieve its mission.

This report describes the algorithms used in DC-SIM. It also describes important features under development, including hyper-real time performance for effective prediction, and an adaptive tuning mechanism for improving fidelity. Also described are various validations of its

Manuscript approved June 19, 2001.

performance, including comparison with the well-known CFAST algorithms [Portier, et.al., 1992; Bailey, et.al., 1995] that were developed primarily for simulation of fire-spread in office buildings.

The report is organized as follows: Section 1 introduces the projects, analyzes their requirements and then briefly discusses project planning; Sections 2 and 3 discuss data abstraction and simulation algorithms in detail; Sections 4 and 5 introduce the computer and user interface of the simulator; Section 6 concerns the scenario generator that is designed and implemented to simplify use of the simulator; Sections 7 and 8 discuss validation of the implemented functions and possible expansion; Appendices A through C provide detailed information on system implementation and usage, and Appendix D provides a comparison of DC-SIM with the Consolidated Model of Fire Growth and Smoke Transport (CFAST). For readers interested in how to use SC-SIM, Sections 4, 5, and Appendix B constitute a simple users manual.

1.1 Project Requirement and Plan

In this section, we briefly introduce the requirements for the simulator imposed by two projects in the domain of ship damage control.

1.1.1 DC-TRAIN Requirements

A major requirement of DC-TRAIN is to collect information about ship status and display it in a centralized way to a DCA, in conjunction with the training of a DCA. The purpose of using DC-SIM in this project is to provide a realistic damage control simulation with real-time performance. The simulator serves as the virtual ship platform for damage control training. Based on training requirements, the simulator needs to handle the following tasks:

- Simulating ship damage for any ship in DC-SIM data format
- Providing a detailed model of ship structure
- Simulating interaction among ship parts
- Real-time simulation and tracking of compartment temperature, pressure, air composition, water depth, pipe pressure, and device operating condition
- Real-time tracking of the consequences of operating any modeled ship element
- Providing an easy user interface to issue damage control commands

1.1.2 DC-SCS Requirement

DC-SCS is a project focused on automatic or computer-aided decision making in damage control. Physical events in the simulation or on ship constitute the basics of upper level intelligent control processes. For this reason, DC-SIM has important applications in the following areas:

- DC-SIM works as a test bed for the decision-making modules. Since it is not feasible to conduct extremely large numbers of tests in a real ship environment, it is crucial to test the system on a virtual ship that is provided by DC-SIM. The test purpose is two fold: features and functions. A feature test checks if the system is operational--for example, does the system receive a high temperature reading? A function test checks if the system makes the correct move--for example, when a compartment is ignited, does the DC-SCS recognize the crisis? The requirement to conduct the first type of test is basic for software engineering, while the requirement of the second type of tests is similar to the DCA's except that it requires more careful calculation of ignition.

- DC-SIM provides examples for improvement to intelligent control algorithms using machine learning. Learning based on trained examples normally requires an extensive number of real examples. Since the cost of creating various damage control situations on a real ship is very high, such as putting the ship in the state caused by a major explosion, a simulator can reduce the cost and increase the speed of the development process dramatically. In order to perform this task well, DC-SIM should be able to create any kind of crisis with reasonably good fidelity. Additional to the requirements previously discussed, DC-SIM should be able to predict correctly the trend of the crisis's spread, to describe the pattern of a crisis's start with good accuracy, and also to provide a mechanism to facilitate on-line and off-line learning.
- Intelligent Control in DC-SCS doesn't have a complete physical model of the control object. Though it may have part of the ship's dynamic characters modeled implicitly in its internal data presentation, there is no proof that the presentation is complete and effective. Therefore DC-SIM, working as an explicit physical model of the control object, can be very helpful to DC-SCS in controlling ship damage:
    - *Hyper-real time crisis prediction.* In conducting casualty control with limited resources, it is vital to order crises based on their seriousness and thus come up with an optimal solution with minimum loss. With hyper- real time performance, DC-SIM can almost instantly make short-term predictions of a crisis's development and spread under various assumptions of intervening casualty control response. The prediction it makes is then provided to the artificial intelligent (AI) decision maker to evaluate the seriousness of the crisis and the effectiveness of casualty control responses available. This usage requires DC-SIM to have hyper-real time performance in short-term prediction. In order to make this goal feasible, the prediction is allowed to be local, i.e. on a compartment cluster, instead of ship-scale. And this usage doesn't impose requirements on higher fidelity as long as only the trend and order of crisis development and spread need to be determined.
    - *Sensor and actuator validation.* Sensors and actuators may malfunction. Assuming that the probability of a sensor malfunction is low, by making a consistency check of a possibly malfunctioning sensor with other sensors, it can be determined if the suspected sensor is malfunctioning or not. DC-SIM provides vital information to conduct the consistency check. Validation of actuators works similarly. Based on DC-SIM'S prediction of the immediate consequences of an actuator's operation, it is possible to tell whether or not the actuator's operation has been conducted successfully.
    - *Crisis recognition.* It is very important to recognize a crisis before it is fully developed. In this case, DC-SIM again provides a consistency check to

isolate a false alarm. The above two usages both impose requirements on the consistency of simulation results.

- In summary, based on the requirements imposed by the DC-TRAIN and DC-SCS projects, DC-SIM should also be able to create all possible crises to make short-term prediction of a crisis's spread trend within a restricted area in hyper-real time, and to provide consistent simulation results.

1.2 Incremental Development Plan

The development of the simulator is incremental. The approach is to have a fully functional simulator, and then incrementally improve its accuracy by incrementally increasing its complexity. For example, the first fire simulator simulated only bulkhead conduction and simple vent convection, and had no air composition tracking. Subsequently, a more complicated convection model was introduced, internal free convection was implemented, boundary layer effects were taken into account, and air composition was tracked. Every upgrade was based on the previous model and code. No major redesign has been found to be necessary, thanks to the straightforward design of basic objects based on real world abstraction.

2. Knowledge Representation

In this section we discuss how the ship's structure is represented abstractly in the DC-SIM format database. The detailed information can be found in Appendix A. The specifications underwent a number of changes as the model became more and more comprehensive and sophisticated.

All the detailed specifications are illustrated in **Appendix A: DC-SIM Ontology**, which may be referred to for questions that are not addressed or answered by the text in this section.

2.1 Representation of Ship Static Structure

A ship consists of compartments, pipe networks, electrical networks, and devices attached to various networks.

- *Compartments*: A compartment consists of a set of bulkheads. Once the bulkheads forming a compartment are specified, the compartment is specified. For vents like doors, we regard them as associated with bulkheads, i.e., a door is inside a bulkhead and therefore identifying the bulkhead also identify a door's location.
    - *Bulkheads:* A bulkhead consists of two sides. Here we require that any bulkhead, actually a bulkhead segment, should have rectangular or triangular shape, and its two sides should be of the same geometry. Thus a bulkhead can be specified by its two sides. The distance between these two sides is the bulkhead's thickness. We assume that all the bulkheads have uniform heat conductivity.

    - *Sides*: A side has a rectangular or triangular shape. It has no thickness. Its geometry is specified by its vertices.
    - *Vertices:* This is the most fundamental element of ship structure. A 3-D coordinate system is associated with the ship. The z-axis goes upward, the x-axis goes forward, and y-axis goes to starboard. For detailed settings, refer to **DC-SIM Ontology** (Appendix A).

- *Pipe networks*: There are three kinds of pipe networks, namely fire main, chill water, and ventilation systems. The ventilation system may not be actually a pipe network, but we approximate it in this way to get a uniform structure and simulation algorithm, and to reduce coding and maintenance effort. However, we need to point out here that using the same pipe network solver on the ventilation network as that used in solving fire main and chill water systems can be a problem because air is compressible while water isn't. But since the ventilation system normally works at low pressure, we regard this a minor problem, and the amount of error it can introduce is within the project's tolerance.
    - *Fire main:* The fire main network is where seawater comes from to fight a fire. Water is pumped into the network at elevated pressure by main pumps. A graphical structure is used to model the fire main and other pipe networks. Basically a fire main consists of pipes, pumps and connections. Pipes and pumps are regarded as edges, and connections as nodes. A node is simply an abstract point, with additional fields indicating type (junction, T-connection or elbow). A pipe is specified by its two end nodes and its diameter. A pump is specified by its two end nodes, its maximum flow rate and its maximum head pressure. Additional dynamic fields also exist, such as operation status and power level. Pipe ruptures are modeled by special types of pipes and nodes. Detailed information can be found in the DC-SIM Ontology section (Appendix A).
    - *Chill water*: The chill water network is the cooling system for ship devices. Its basic structure is the same as the fire main's, except that it doesn't get water from the sea. It is a closed pipe network with no connection to the environment (except when there is a rupture, or certain discharge valves are open). Its specification is thus the same as that of fire main, except that it has heat exchange devices built in for cooling purposes.
    - *Ventilation system*: The ventilation system consists of fans and ducts. It can be viewed as a pipe network structure.
- *Electrical network*: This is the power source of the whole ship. The electrical network contains generators, circuits, and various devices. It's not fully in the scope of this simulation design. We are only interested in the heat generating properties of its devices, and its exact topology for damage injection.
- *Various devices*: This is a broad area, including systems from portable pumps to radar. Most of them are modeled in the networks.

2.2 Representation of Ship Dynamic Status

Ship dynamic status includes any physical properties, geometrical characters and device operation status.

2.2.1 Physical properties

Physical properties are the major task of simulation. The majority of them are contained in the Compartment Status table, including zone heights, zone temperatures, zone pressures, detailed air composition, combustible fuel amount, suppressed fuel amount, and water depth. Others are in tables such as Fire Main Pressure and Fire Main Flow Rate.

2.2.2. Geometrical characters

These are status of ship's geometrical structure. Bulkhead ruptures, vent (doors and hatches) status and pipe ruptures are included. They are contained in corresponding tables like Wall Rupture table, various vent tables, and Fire Main Rupture table.

2.2.3 Device operation status

These are the operational status of various devices, including pumps, valves, sprinklers and all the electrical devices. They are included in tables of Fire Main Pumps, Fire Main Valves, Fire Main Sprinklers, and those for electrical devices.

3. Simulation Engine

So far we have introduced the data representation of ship damage control simulation in the central database. The simulation engines constitute the core of the simulation. They operate directly on the data representation. More specifically, based on the problem setting, a simulation engine reads in the initial state, simulates the continuous state change, and at the same time reports it to the central database. (Refer to Section 4: **Simulation Computer Interface.**) For example, the fire main simulation engine calculates pipe network flow rate and pressure distribution, and monitors network status change. Once some operation like opening a valve occurs, it updates the data it keeps, solves fire main again if necessary, and then reports the new flow rate and pressure distribution to the central database for the simulation user's retrieval. The following depicts what can happen in a compartment:

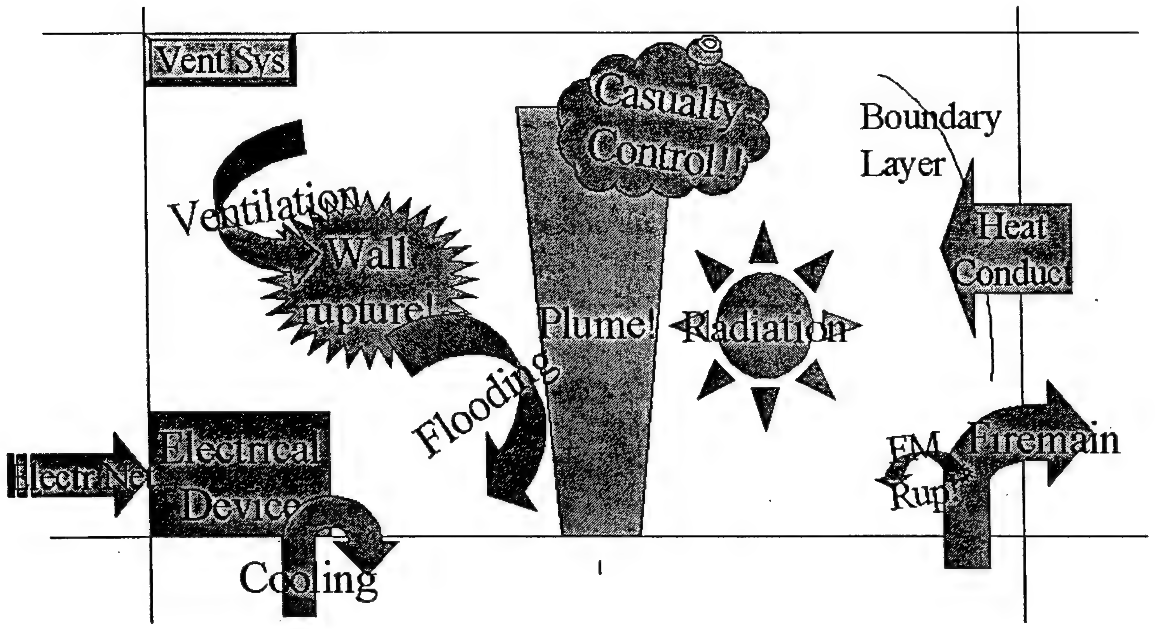


Figure 1. What can happen in a compartment

First, there can be combustion inside a compartment. This is represented by the plume in the center of the Figure. Driven by the pressure and mass concentration gap, convection occurs at doors and bulkhead ruptures. So does internal convection, i.e. hot air up and cool air down. And driven by a temperature gap across a bulkhead segment, conduction occurs. Both affect and are affected by the boundary layer. The plume can also generate radiation. These are roughly what should be simulated by the fire simulation engine.

Second, flooding can be introduced, either a fire main rupture or simply from neighboring compartments or the sea. And there can be interaction between fire and flooding: either flooding puts out fire, or fire spreads with flooding. Flooding propagation should be handled by a fire main solver and a flooding simulation engine, and its interaction with fire by part of fire simulation.

Third, the electrical devices in the compartment can also cause problems; for example, fire caused by shorting. This is coupled with the cooling system, i.e., chill-water, and also with the rest of electrical network. This part should be handled by an electrical network solver and a chill water system solver.

Finally, there is the casualty control system: watermist, AFFF, fire boundary, de-flooding or de-smoking. These are distributed among various simulation engines.

Though the design is compartment based, all the compartments can fit together seamlessly to form a global view of the whole ship.

In the following sections we discuss the simulation engines one by one in a fairly detailed manner.

3.1 Fire and Smoke Simulation

This is the most important and complicated part of DC-SIM. The main task is to predict the continuous status change of a group of compartments affected by fire. This means that temperature, gas concentration, pressure, fuel remaining, and other parameters for any compartment affected by fire are traced in real time, and that concurrent events such as ignition, casualty control response, and interaction with other modules like flooding are simulated concurrently.

The overall design of the fire simulator is object oriented. We didn't adopt methods used in large-scale numerical simulation coding that on the one hand typically provide fast numerical calculation speed and are easy to optimize from a numerical analysis perspective, but on the other hand are somewhat incompetent in representing real-world objects, less intuitive, and prone to introduce bugs during coding. Rather, we model each compartment with its internal physical and chemical structure, and also model each compartment's interaction with compartment-connections such as bulkheads or doors. So, putting the compartments together automatically forms a complete image of the ship. Working in this way, it is fairly easy to add new features. For example, what if we want to add simulation of a fire boundary? There is no need to get global representation and control of all the fire boundaries set up on the ship. Instead, a fire boundary just maintains the side temperature of a bulkhead, and has some effect on water vapor concentration and water depth (if necessary) of the compartment that it is set in. So we can say the design is "localized". Every feature is kept in an appropriate object locally, while their structural sum forms a whole ship with integrated functions.

3.1.1 Mathematical Model

Simulation of heat transfer with air motion can be accomplished in several ways with differing speed, fidelity, and adaptability. A relatively simple model, the two-zone model, is used in DC-SIM for fire and smoke emulation. This choice is based on two facts: First, a rough estimate of computation requirements indicates that using more sophisticated models may not be able to render real time performance; second, experiments have illustrated that a compartment on fire typically has two zones with relatively uniform physical properties.

The mathematical model for fire and smoke can be divided into several sections, each of which deals with a specific phenomenon. We start with the combustion process from which mass and energy is produced.

3.1.1.1 Combustion

Combustion is the ultimate force for most heat transfer and air motion. Theoretically, a combustion process is determined by:

- Chemical properties and surface condition of the combustible;
- Temperature, pressure and air composition around the combustible;

Based on these conditions, thermodynamics determines the chemical reaction type and combustion speed.

However, such a detailed model is far too complicated to maintain at real time on a personal computer, and it doesn't fit our project's requirement either. Rather, we use approximation to greatly reduce the computational complexity:

- Reaction types, or from another point of view, reaction equilibrium, is assumed to depend on a compartment's average oxygen concentration only. For example, when carbon is burnt, both carbon dioxide and carbon monoxide can be produced by two different processes, i.e., complete and incomplete burning. We assume that the relative ratio of these two combustion processes depends on the oxygen concentration only. This is a very rough assumption since temperature and environment gas concentration can also affect the process.
- Combustion speed is assumed to be proportional to oxygen concentration. In this way, we disregard the effect of temperature and fuel surface condition.

Based on these simplifications, the combustion can be simulated effectively using fuel characterization provided by the central database. The central database contains two tables related with fuel properties: FuelDescription and ReactionProperties. The FuelDescription table contains basic data on fuel consumption rate, heat released by complete/incomplete burning of a unit of the fuel, and also soot released by complete/incomplete burning of unit mass of the fuel. The ReactionProperties table contains description of possible reactions. It actually describes the equation for each type of reaction. The coefficients of reactant are negative, and products positive, both normalized to consumption of one mole of fuel[1].

3.1.1.2 Radiation

Due to the computational complexity that would be introduced by geometrical calculation of radiation, we make the simplified assumption that a compartment has just two uniformly distributed zones, and we don't simulate radiation in detail. Instead, we simulate plume radiation by an empirical factor that is tunable in the user interface. (See Section 5.1). The default value is set as 0.8 which means that 80% of the heat is assumed to be turned into radiation, and distributed into the two zones evenly based on zone heights.

The lack of simulations of other types of radiation brings numerical error. Fortunately, part of the error is compensated by the basic two-zone model we use, and by the parameter tuning mechanism.

3.1.1.3 Internal free convection

Normally in a compartment affected by fire, the temperature of the lower zone is lower than that of the upper zone. This phenomenon results from hot air's lower density than cool air's, and is coupled with the boundary layer's motion.

In this project we don't plan to solve the boundary layer equation accurately. Therefore we also used an aggressive approximation to simplify interval free convection simulation. The goal is to reach a reasonable temperature distribution between the upper and lower zone without too much

[1] This table may also contain heat and soot information. This is redundant and actually not used by DC-SIM.

effort in both coding and computing. The method we adopted is as following: Based on the temperature gap between the upper and lower zone, a certain amount of upward and downward airflow is calculated and it changes zone heights. This is a dynamic compensation process that modifies each zone's height and temperature. This process is not launched until a compartment is affected by fire, and it eventually reaches a certain equilibrium that can be empirically tested. In designing this process, there is no guarantee that its result will be realistic, but we believe that its structure, i.e. the dynamic compensation mechanism, will provide realistic results if its parameters are chosen wisely. These parameters can be tuned in a user interface at real time when simulation is running. (See Section 5.1).

3.1.1.4 Vent convection

The convection through an open vent is simulated by solving a Bernoulli equation with a restriction factor.

Only convection between the corresponding upper zones and lower zones are calculated. The resulting flow rate is then used to calculate a compartment's mass distribution, temperature distribution, and change of gas concentration.

Since the simulation step is fixed for compartments with different size, fast convection can introduce numerical instability. To solve this problem we used a factor determined by a compartment's geometry to restrict the airflow rate through any vent connected to this compartment. Though this method inevitably affects the fidelity of simulation, its overall performance seems to be good: it eliminates data oscillation while still maintaining realistic fire spread speed.

Special phenomena like plumes shooting from open vents are not simulated exactly. In these cases, the simulator may not be able to give an accurate result in a small time span. But from a perspective of long-term simulation, this efficiency doesn't affect simulation accuracy.

3.1.1.5 Bulkhead conduction

Bulkhead conduction is determined by a one-dimensional Fourier equation. Since a compartment has two zones with different temperature, a bulkhead's two sides also have their separate temperature zones. Therefore a bulkhead segment is normally cut into 3 parts, each of which has uniform temperature at either side. Heat conduction is calculated with each part. The conduction calculation is relatively simple. Since insulation, composition and thickness vary for bulkheads, we give users the choice to tune a coefficient to adjust bulkhead conductivity. (See Section 5.1). We must point out that although the conductivity is allowed to be tuned, it is only recommended to do so if the user believes that the default conductivity is not appropriate.

Bulkhead conduction simulation has been compared by CFAST based on a one-compartment case. (See Appendix D).

3.1.1.6 Boundary layer effect

Bulkhead conduction and free convection both contribute in forming a layer of air with a temperature gradient. The layer has uneven thickness, and is in constant motion that contributes to free internal convection as discussed in the previous section. In our two-zone simplified model, we don't try to solve the boundary equation explicitly. Instead, we approximate its effect on heat and mass transfer by:

1. Heat transfer on bulkhead conduction: Since the two-zone model assumes uniform temperature distribution in either zone, the temperature gap across a bulkhead segment would be too much if we just use the zone temperature gap. In order to correct this problem, we introduce a factor called Boundary Layer Coefficient to reduce the temperature gap across a bulkhead segment. So by tuning this parameter, the effect of boundary layer on conduction can be regulated. (See Section 5.1).
2. Contribution to heat and mass transfer in internal free convection: This has been taken into account in the section on internal free convection; i.e., the effect has been covered by the mechanism we established to simulate internal free convection and therefore here we don't need an additional structure to simulate the boundary layer effect explicitly.

3.1.1.7 Direct fire fighting

Simulation of fire fighting is part of DC-SIM'S objectives. DC-SIM can do the following simulation in an empirical way:

Water Mist, seawater, AFFF, Halon, and CO_2. Water reduces fire size, changes fuel's combustibility, and decreases temperature when vaporizing. The exact amount of water that is vaporized and applied to fuel cannot be determined explicitly. Therefore only a very rough estimate is available. In DC-SIM, we use "dose" to indicate the amount of effort that is applied. 100 dose stands for the maximum, and 0 for the minimum. The relationship between fire control performance and dose can be tuned by experienced users. AFFF, Halon and CO_2 are simulated in similar ways, except that AFFF affects only fuel combustibility, Halon only oxygen concentration.

3.1.1.8 Fire boundaries

Fire boundaries can be set on any side of a bulkhead segment. By setting a fire boundary on a side, that side's temperature is maintained at a constant level, and thus further heat transfer to the compartment that is protected by the fire boundary is deterred (depending on how completely the compartment is protected by fire boundaries). Failure to maintain a fire boundary has not yet been implemented.

3.1.1.9 Ventilation (de-smoking)

The ventilation system consists of a set of ducts and fans. The structure of the system is similar to that of fire main. But this doesn't indicate that we can use the same numerical algorithm. Simulating ventilation is more complicated since air is compressible while water isn't. Actually based on our project requirement, it is impossible to solve the ventilation system in detail at good fidelity. So a compromise has to be made, which is just one like the fire main simulation, together with a monitor on normal air pressure to make sure that the air can be regarded as incompressible. Such a simulator for the ventilation system can be anticipated in the near future, but currently DC-SIM uses only a rudimentary method to simulate de-smoking. Upon receiving a de-smoking command, a certain flow rate (both in and out) is assumed and the air composition re-calculated. De-smoking can cause re-ignition as it may raise oxygen concentration.

3.1.2 Implementation

The fire and smoke simulation engine is implemented in class CFire. For more information, please see Class CFire in Appendix C.

3.2 Flooding Simulation

The main task for flooding simulation is to trace water flow through various vents and water levels inside any flooded compartments. Ideally ship listing should also be considered, as it is very important for ship survival and affects water flow among compartments. But for simplicity we assume that effective counter-flooding measures are always carried out in time. So the ship is assumed to be in horizontal position all the time.

3.2.1 Mathematical Model

The water surface inside a compartment is assumed to be horizontal. Therefore a water height gap occurs across open vents. This is taken as the driving force of water flow through vents. In detail, Bernoulli's Equation with restriction factor modeling energy loss in friction and in other forms is used.

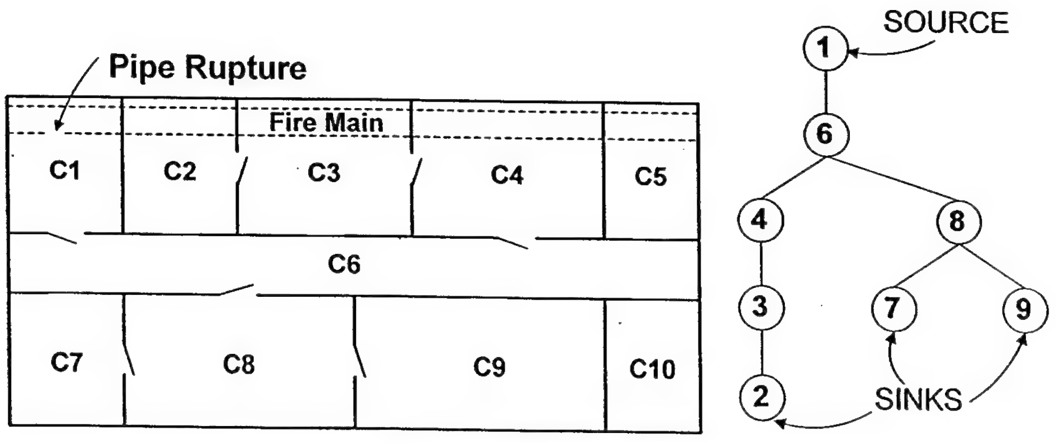


Figure 2. Modeling flooding

Applying Bernoulli's equation to two adjacent compartments that are not full of water is easy since the surface pressure of each compartment can be traced. But in case that one compartment is full of water, its surface pressure cannot be determined by itself, nor can it be determined by looking at its neighbors. Instead, the surface pressure is directly affected by all the compartments that are full of water and by those to which it is connected. For example, suppose compartment A is below compartment B and C, and A and B are connected by an open scuttle, while the scuttle connecting A and C are closed. If B is flooded and A is full of water, A's surface pressure is determined by B's surface pressure, assuming water flow doesn't involve too much head loss. But if suddenly the scuttle connection between A and C is opened (or destructed by some explosion), there can be an abrupt change of A's surface pressure and that can also affect B's surface pressure if B's also full of water. Therefore, in order to have correct surface pressure data, it is necessary to keep a global picture of the compartments flooded, especially all the compartments that are full of water. This picture needs to be analyzed and updated in every round of simulation.

Unfortunately, maintaining the global picture is not a good choice since it violates the object oriented design and localization principle. Doing so would introduce heavy overhead once a ship structural change occurs, and even instability if two water sources are competing to determine the surface pressure of a compartment. In order to localize this problem, while at the same time not introducing too much complexity into the simulator, we assigned a virtual pump (residual water) to each compartment. The pump actually stands for the very slight compressibility that water can have. The mechanism works in this way: When a compartment is full of water, it can still hold a tiny amount of residual water. The residual water forms residual pressure and the residual pressure, together with a temporary surface pressure, can determine in which way water flows even when a compartment and all its neighbors are full of water. Once this is determined, the temporary surface pressure can be updated based on the direction of water flow. In this way, we just introduced very little computational overhead, but saved the trouble in tracing the whole graph of the compartments that are connected to find out the correct surface pressure.

When examined closely, this residual water method cannot promise that the amount of residual water, and the direction of water flow, will remain stable. This is true in some well-chosen cases. But, such local instability doesn't affect over fidelity of flooding simulation. It turns out eventually (and actually very soon) the process reaches equilibrium and the total amount of residual water is very small.

3.2.2 Implementation

Some information on flooding implementation can be found in Appendix C.

3.3 Fire Main Simulation

The fire main is the pipe network to provide water for fire fighting purposes. Its water resource is the sea. This network covers the whole ship and is powered by the fire pumps.

The simulation's purpose is based on its configuration to find out the pressure and flow rate at any point of the fire main. Pressure is regarded as what can fight a fire in the damage control domain and flow rate, especially flow rate at a fire main rupture, is a common and serious cause of flooding.

3.3.1 Mathematical Model

The basic equation of flow in pipe systems is as following, which is essentially an equation on **energy conservation**:

$$P_1 + \tfrac{1}{2}\rho V_1^2 + \rho g h_1 + \rho g h_p = P_2 + \tfrac{1}{2}\rho V_2^2 + \rho g h_2 + \sum \tfrac{1}{2} K_l \rho V^2 + \sum \tfrac{fL}{2D}\rho V^2$$

From the left to the right of the above equation:

- The first term is the pressure at point 1;
- The second term is the kinetic energy (in pressure units) at point 1;

- The third term is the potential energy (in pressure units) at point 1;
- The fourth term is the energy provided by a pump which pushes water from point 1 to point 2, and hp is the head of the pump;
- The fifth, sixth, and seventh terms are the pressure, kinetic energy, and potential energy at point 2 respectively;
- The eighth term is the sum of all fitting loss. A **fitting loss** is basically a kind of energy loss due to flow friction caused by pipe expansion/contraction, bend, valves, gauges, and Tee connections. Kl is a **fitting loss coefficient** at a specific point between point 1 and point 2, and V is the flow rate at that point;
- The ninth term is the sum of all head loss caused by friction inside flow and between the flow and pipe wall. f is a **friction factor** of a segment of pipe where the pipe's material (coarse or smooth) and diameter are constants. L is the length of the segment, and D is the diameter. Since the diameter doesn't change, the flow rate in that segment is also a constant, which is exactly the V in this term.

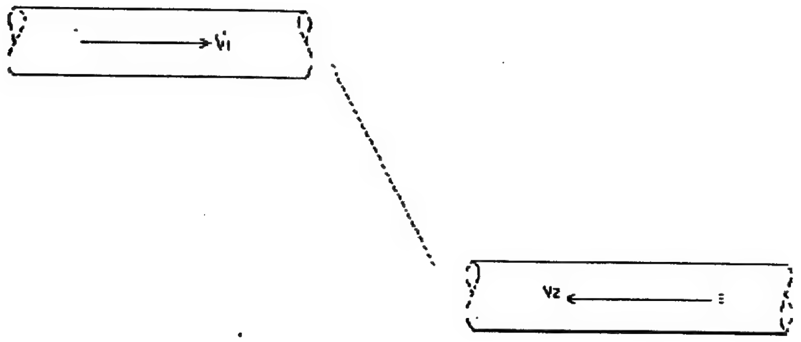


Figure 3. Basic equation for water flow in a pipe

The companion equation is derived from the **incompressibility of water**. Generally, for any point in the pipe system, the in-flow rate and out-flow rate are the same, namely, their algebraic summation is 0:

$$\sum VA = 0$$

In order to solve the above equations for flow speed V (or flow rate Q=VA), further information on fitting loss coefficient Kl and friction factor f are needed. Kl is determined empirically, while we do have specific equations to solve f based on flow status.

A flow may be laminar, or turbulent. The different flows have different friction factors. The status of flow is judged based on a constant called **Reynolds** that is defined as following:

$$\mathrm{Re} = VD/\gamma$$

where

$$\gamma = \mu/\rho$$

is the **kinematical viscosity** of water , μ the **dynamic viscosity**. Reynolds is a physical parameter without unit.

Generally if the Reynolds is greater than 2000, the flow is **turbulent**, else it is **laminar**. 2000 is an empirical number and may be adjusted downward if the flow is in a highly unstable environment (such as a ship's fire main) because disturbances such as vibration make it much easier to cause turbulence. Actually in this simulator we choose 1000, and that can be tuned based on the exact ship condition. If the flow is laminar, the friction factor can be calculated directly.

But if the flow is turbulent, we need to solve an equation:

$$1/\sqrt{f} = 2\log(\mathrm{Re}\sqrt{f}) - 0.8$$

A pump on line also has an effect on the flow. A pump is described by its **system curve.** A typical system curve is as following

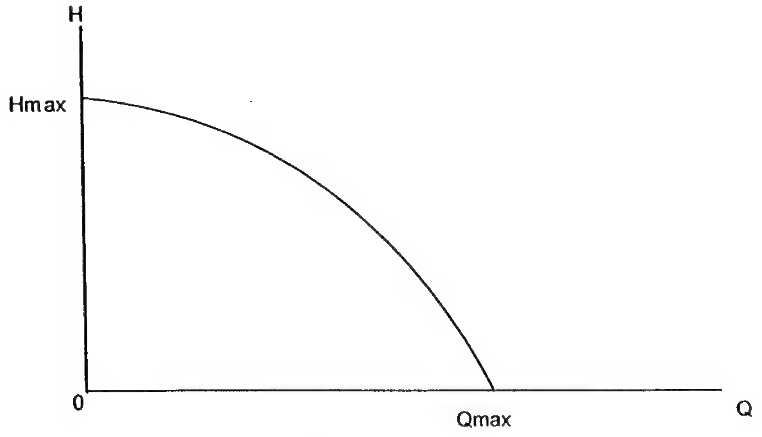


Figure 4. Pump model

where Hmax is the maximum head gain, Qmax the maximum flow rate. The system curve is a 1-1 correspondence between flow rate and head gain. The curve may not be quadratic, but if no further information is available except Hmax and Qmax, it is generally approximated by a quadratic equation:

$$h_p = H_{\max}(1 - Q^2 / Q^2_{\max})$$

Theoretically, based on the equations described above, any pipe network can be analyzed numerically. But for the sake of simplicity and as a comparison with those equations used in the flood module, we mention the Darcy-Weisbach equation here:

$$h_l = (8fL / g\pi^2 D^5) Q^2$$

This equation relates head loss to flow rate directly. Essentially it is equivalent to the basic equation mentioned at the beginning of this subsection.

3.3.1.1 Numerical Method

Equations governing the flow and pressure distribution along the fire main are nonlinear (see part 2). The numerical algorithm we use is based on the fact that there are two criteria that a balanced flow in a network must satisfy:

1. The net flow into any junction must be zero. This means the flow rate into the junction must equal the flow rate out of the junction.
2. The net pressure raise/drop (or head gain/loss) around a loop must be zero.

(Recall that in circuit theory, current and voltage also satisfy these conditions. The difference here is that the relationship between flow rate and pressure is nonlinear.)

Now suppose we have a flow rate distribution satisfying condition one. If the net head loss is also zero, then we're done. The flow rate distribution is balanced. If the net head loss is not zero along each loop, the current flow rate distribution needs to be modified. The procedure is essentially the same as Newton-Raphson's method in solving nonlinear equations and is developed as following:

In a given loop, supposed Q_0 is the balanced, natural flow rate and Q the current flow rate. Let f(Q) be the net head loss along the loop. We have

$$f(Q_0) \equiv 0$$

By criteria 2. By Taylor's expansion,

$$f(Q_0) = f(Q) + \frac{df(Q)}{dQ}(Q_0 - Q) + \ldots$$

Retaining only the first order term in the expansion, we have

$$Q_0 - Q = -\frac{f(Q)}{df(Q)/dQ}$$

This is a reasonable modification on current flow rate Q.

Practically f(Q) is expressed by Darcy-Weisbach's equation and the equation of the pump's system curve.

The algorithm works as an iteration until f(Q) is small enough.

3.3.2 Data Abstraction

In order to discuss the treatment of rupture, at first we need to clarify basic concepts in describing the fire main in the simulator.

3.3.2.1 Unit/Edge of the fire main

Basically this is a pipeline in a compartment. But in the following cases, a compartment may have more than one unit.

- Pipe expansion/contraction. If two pipes of different diameter are connected in a compartment, they are regarded as two units.
- Tee connection. Three pipes are connected together. Each pipe is regarded as one unit.
- Pumps. If a pump's input and output pipes have the same diameter, the pump and these two pipes are regarded as one unit. Otherwise the input pipe is one unit, and the pump with the output pipe is another unit.
- Valves and gauges. If there is a valve or gauge in a pipeline, the line is cut into two units.

Combined with some topological descriptions, a unit is declared as an **edge**.

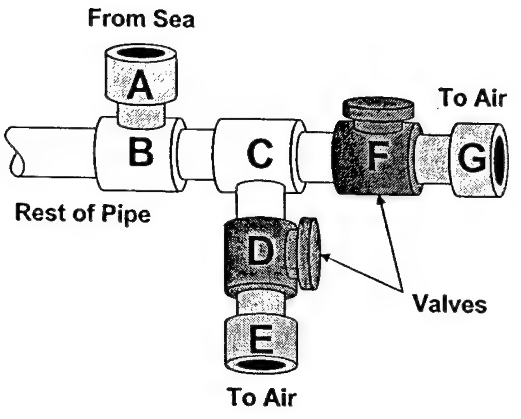
3.3.2.2 Node of the fire main

This is an end of any edge, i.e. a pipe expansion/contraction, a Tee connection, a valve, a gauge, or an intersection of pipe and bulkhead/deck/overhead. A node is an abstract point.

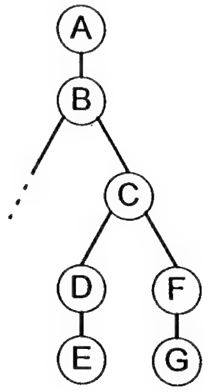
3.3.2.3 Basic implementation of the fire main:

We implement the fire main as an Undirected Simple Graph.

The following figure is an example of data abstraction.



(a)



(b)

Figure 5. Data abstraction of rupture module

3.3.2.4 Default flow direction

The default flow direction in an edge is used as a reference of the real flow direction. It can be chosen arbitrarily, except that if an edge contains a pump, the default direction should be the same as the direction the pump is pushing the water.

3.3.2.5 Reported flow rate

This is a signed number. A positive sign means the flow is going in the default direction, and a negative sign means the flow is opposite the default direction.

3.3.2.6 Default reported pressure

This is the pressure at a node. If a node is a closed valve or any other node type with different pressure at different sides, each pressure will be reported separately.

Based on the flow rate and default reported pressure, the pressure at virtually any point of the fire main can be predicted.

3.3.2.7 Powered vs. Unpowered Components

The fire main may not be a totally interconnected graph. Sometimes part of the fire main is disconnected from the water source (the sea) as a result of rupture or closed valves. In this case, the isolated part is called the Unpowered Component, and is not simulated. The fire main we actually simulate is the components of the fire main that are connected with water sources, i.e. Powered Components.

3.3.3 Comments on Ruptures

The following are two important issues related to crises.

3.3.3.1 Simulating ruptures

Simulating and detecting a rupture of the fire main is a key goal of this simulator. Essentially, a rupture is an Air node if it is exposed in air, or a Sea node if it is immersed by water. In order to distinguish it from normal virtual nodes such as the water entrance of fire main or an exit like the fireplug, a node describing a rupture has a member called RuptureDegree indicating if it's a rupture. A rupture has 5 different degrees, labeled from 1 to 5. 5 means open rupture, which is the most serious. We add 0 as a rupture degree to indicate 'intact'.

3.3.3.2 Detecting ruptures

The position and degree of a rupture can be detected by this simulator based on such information as pressure distribution reported by pressure gauges and flow rate gauges. The more detailed the information reported by these gauges, the more accurate the detection. In order to find out the exact position and degree of a rupture, the simulator needs to know the pressure at every Tee connection. Therefore, the upper bound on the number of gauges is the number of Tee connections in the fire main. But since normally ruptures are located at the damaged part, the number of gauges can be reasonably reduced, and the simulator is still able to give a quite accurate estimation. Furthermore, by temporarily operating on some remote controlled valves, precious information on pressure distribution is available and it can reduce the number of gauges substantially. The placement of those gauges should be plotted carefully to get optimal effect.

3.3.4 Implementation

Some information about fire main implementation can be found in Appendix C.

4. Simulation Computer Interface

The simulation provides a computer interface for exporting simulation results and exchanging information with simulation users (in DC projects these are AI decision making modules, 3-D visualization module, and program test team members.) The interface includes two parts: MASSCOMP [Street, et.al., 2000] format sensor output and DC-SIM database information exchange. The database interface is implemented using open database connectivity (ODBC) standard exclusively.

In this section, we discuss the computer interface. First we discuss simulation results as output through a MASSCOMP format sensor report and the DC-SIM database, and then discuss information exchange using the Event Communication Language (ECL).

4.1 Exporting Simulation Results: MASSCOMP and the DC-SIM Database

The simulation exports its result in real time. In other words, the central database mirrors changes in ship status based on the simulation's time report. To simplify the use of DC-SIM and also to speed up data reporting, MASSCOMP sensor reporting is provided in text file format.

4.1.1 DC-SIM Database Report

The central database has a complete representation of the ship being simulated, including ever-changing ship status: compartment status and pipe network status (fire main, chill water, and ventilation).

4.1.1.1 Compartment status

Compartment status, including zone heights, zone temperatures, pressure, water height, various gas concentrations, and fuel information, is exported to CompartmentStatus table and CompartmentStatusTrace table for every short fixed interval. The status trace table is an extension of the status table along the time axis, i.e. it contains the status table's most important fields, plus one more field called 'TimeTag' to record the time. To avoid the trace table growing too huge, certain measures are taken not to add identical or similar information. More specifically, DC-SIM keeps an internal record indicating the minimum change required for a new record to be added. For example, if the value of the Pressure field of a record is 5000, this means that a pressure change of 5000 Pa in a compartment will add a new record for that compartment into the trace table. If no variable/field has a value change greater than the minimum change specified in the default record, then no new record will be added. The default internal record can be overridden by a special record in CompartmentStatusTrace table with CompartmentID "–2" (all normal compartment IDs are positive) for the user to determine the granularity of the trace table report.

Reporting to CompartmentStatusTrace table is enabled only in DCA project.

The CompartmentStatus table is updated at a fixed interval. Not all the compartments are updated. The simulation judges from its calculations which compartments' status have been changed in the last simulation interval, and exports new information to the table. The size of this table is fixed, and unlike the trace table, the user cannot interfere with the frequency of reporting to this table.

4.1.1.2 Fire main status

This is another important feature of ship status. The fire main simulation determines pressure and flow rate distribution once fire main configuration is changed, i.e. opening a valve or turning on a pump. The result is exported to the following tables: FireMainFlowRates, FireMainPressures, and FireMainRuptures.

Among the simulation submodules there are interconnections. For example, fire main rupture can cause flooding. These interconnections are accomplished at the intra-simulation thread level, not through the database.

The status of the chill water network and ventilation network are not included in the DC-SIM central database at the current time. They will be in place in the future.

4.1.2 MASSCOMP Report

MASSCOMP output is enabled to report sensor reading as the MASSCOMP system on the ex-USS *Shadwell* does. Text format is used to speed up the output process at run time and to facilitate its use without knowledge of the complicated DC-SIM Ontology being necessary. The following is a sample segment of the output file:

```
<SEQ=33
TIME=00:00:33
B158=19.85
A074=19.85
B157=19.85
B159=19.85
B160=19.85
A073=19.85
A026=19.85
A025=19.85
A078=19.85
A077=19.85
```

Every segment in <> is a block of sensor report at a given time. Here the time is 33 seconds since start of simulation. The left hand side of an entry is the sensor's channel number and the right hand side is its reading. For example, "A074=19.85" means that the sensor with channel number A074 has reading 19.85. Checking the CompartmentSensors or FiremainSensors table shows that the sensor is a temperature sensor and the reading is therefore in degrees Celsius.

The granularity of control is accomplished by the same "minimum change" mechanism discussed in recording compartment status trace. Actually, if MASSCOMP output is enabled, DC-SIM will not report to the CompartmentStatusTrace table at all. Instead, it reports to the SensorInput table and the MASSCOMP format file. This is necessary because in DC-ARM, sensor readings can be from actual sensors on a ship, and a module other than DC-SIM will receive these data and put them into CompartmentStatusTrace table. Since testing the whole system means testing that module as well, DC-SIM must be prevented from reporting to the CompartmentStatusTrace table, in order to avoid collision. So, keep it in mind that ***in the DCA project, reporting to the CompartmentStatusTrace table is enabled; while in DC-ARM, it is not, and reporting to SensorInput table and MASSCOMP text file are enabled instead.***

4.2 Event Communication

The database interface is also designed to facilitate event-driven simulation. Ideally, any change in ship environment should affect the simulation process, and DC-SIM is designed to get the change in time and reflect it in the simulation process that follows.

The Event Communication Language (ECL) is designed for event communication among DCA/DC-ARM project submodules. DC-SIM uses this protocol for event communication. For example, an operation of a door that results in change of the door status is an event sent by some DC-SIM user. By monitoring the ECL table, the ECL communication center, DC-SIM can retrieve any simulation-related events in real time. For example, once a door status change event is recognized, the simulation updates the door status in its internal ship representation, and the following simulation is based on the new door status.

For detailed ECL protocol specification, see **Appendix A: DC-SIM Ontology** and **Appendix B: Examples of Using Various Simulation Features.**

5. Simulation User Interface

An MDI interface is used for DC-SIM. The main function of this user interface can be classified into three groups: parameter tuning, simulation status monitoring, and auxiliary functions.

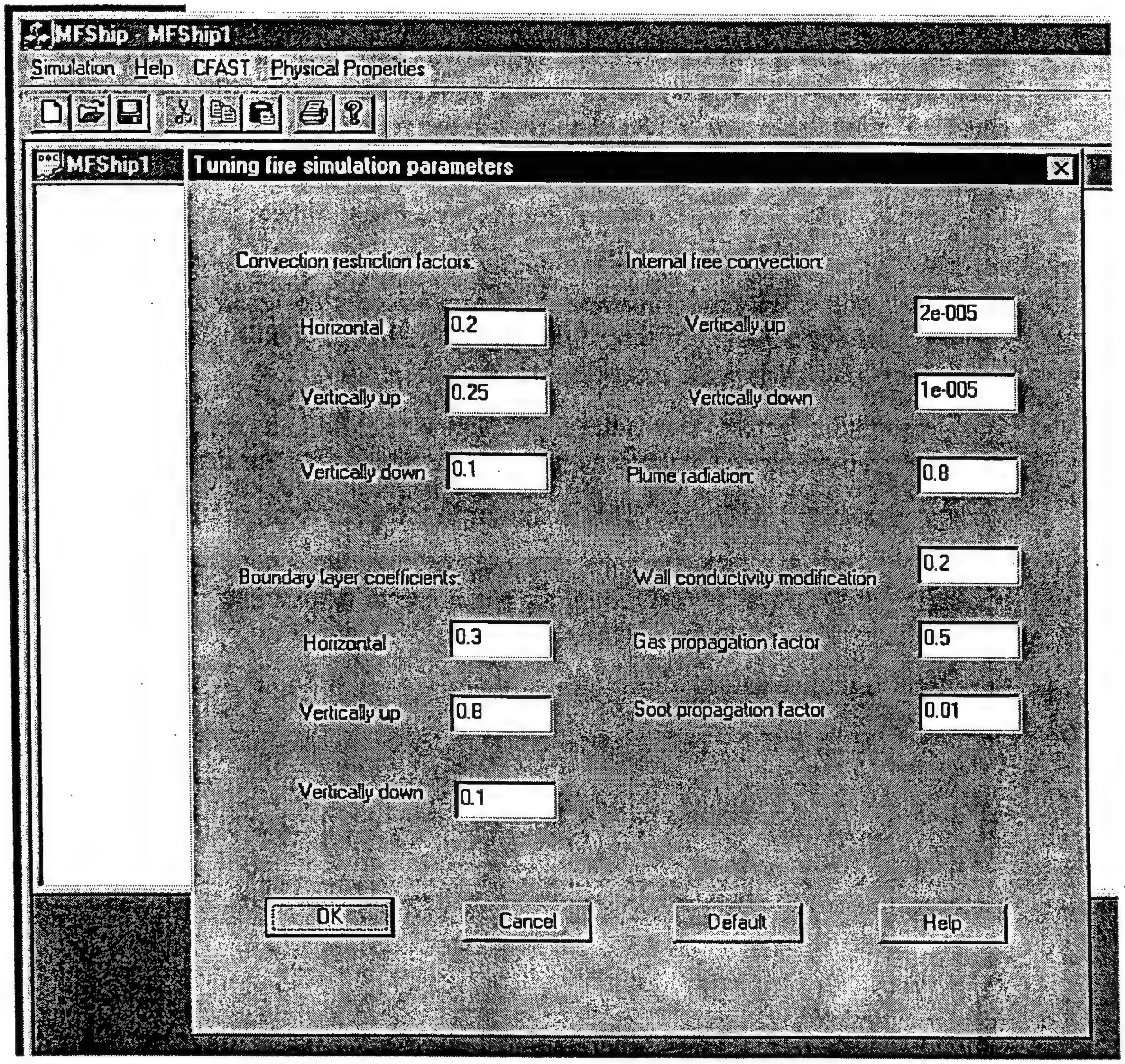
5.1 Parameter Tuning

Parameter tuning is another way to communicate with the simulation. Unlike the database interface that mainly provides control of simulation by changing ship status, the user interface provides control of simulation by changing basic simulation parameters.

As discussed before, one of DC-SIM'S fundamental assumptions is that "exact" real time solution of differential equations governing the underlying physical and chemical processes is impossible given the computational resources available and the real time ship-scale simulation requirement. So a conclusion is made that approximation is necessary and the simulation engines discussed in Section 3 are equipped with simplified algorithms with empirical parameters that can be tuned. The user interface provides a parameter-tuning mechanism that can convey the user's choice to the DC-SIM core any time after DC-SIM is launched.

Figure 6 shows the parameter-tuning dialogs. The appropriate dialog is popped up when the user chooses the set of parameters they want to tune from the main menu: Simulation->Tune Simulation Parameters->Tune fire/flooding/… Parameters.

5.1.1 Tuning Fire Parameters


MFShip - MFShip1
Simulation Help CFAST Physical Properties
MFShip1
Tuning fire simulation parameters
OK
Cancel
Default
Help

Convection restriction factors:

Horizontal	0.2
Vertically up	0.25
Vertically down	0.1

Boundary layer coefficients:

Horizontal	0.3
Vertically up	0.8
Vertically down	0.1

Internal free convection:

Vertically up	2e-005
Vertically down	1e-005
Plume radiation:	0.8
Wall conductivity modification	0.2
Gas propagation factor	0.5
Soot propagation factor	0.01

Figure 6. Fire parameter tuning dialog. This dialog shows the tunable parameters related to fire simulation and their default value

Here is a brief explanation of each fire parameter. To fully understand how it works, the user should refer to Section 3.

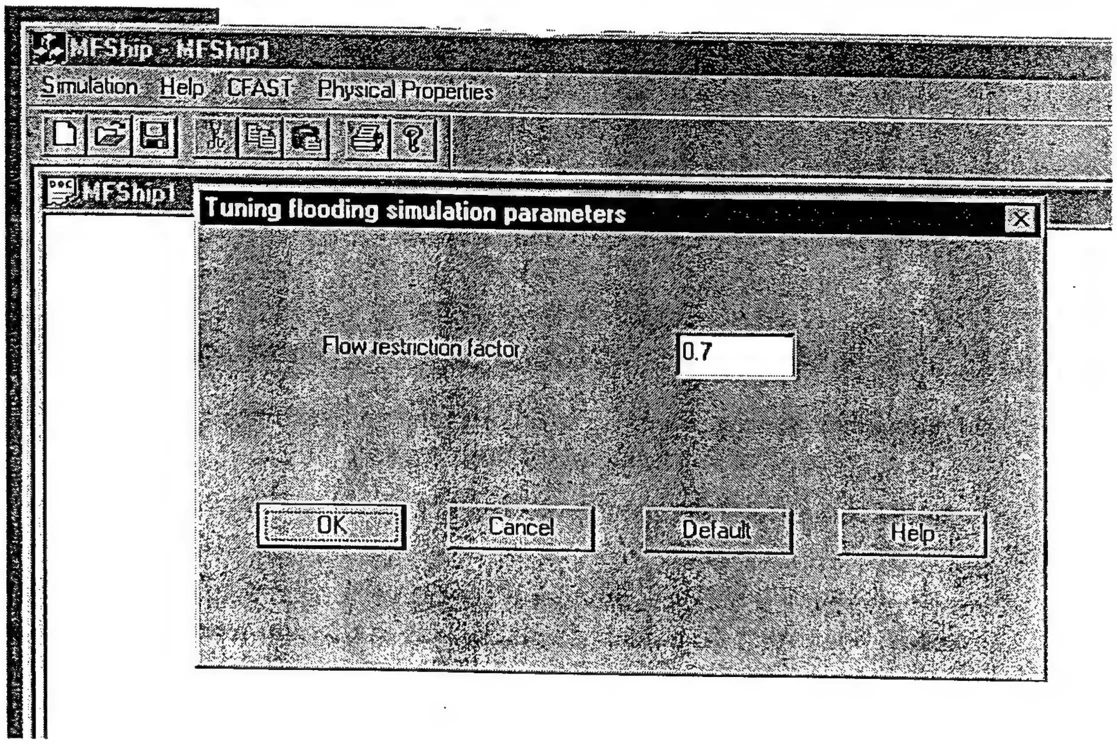
- Convection restriction factor: Bernoulli's equation is used in calculating airflow through an open vent. To model energy loss due to friction, restriction factors are used and they can be tuned here. Two coefficients are used for vertical flow to tune the fire's downward and upward spreading speed. Increasing the coefficients reduces energy lost in the form of friction and thus speeds up convection.
- Boundary layer coefficients: These coefficients are used to in a mechanism to approximate the boundary layer effect (Section 3.1). Again, two coefficients are used,

one for the boundary layer under a deck, and the other for that above a deck. Tuning these two coefficients can result in the downward and upward fire spreading speed the user wants. Increasing these coefficients reduces the boundary layer effect and thus speeds up heat transfer.

- Internal free convection coefficients: These two coefficients are used in calculating internal free convection, i.e., hot air up and cool air down. DC-SIM uses a two-zone model. Tuning these two factors can result in the zone temperature balance the user wants. Increasing these coefficients increases internal free convection.
- Plume radiation: This factor determines the percentage of heat that is radiated from a plume.
- Bulkhead conductivity modification: This coefficient is designed to correct bulkhead conductivity that affects fire spreading through bulkhead conduction. A multiplication correction is applied on the actual bulkhead conductivity.
- Gas propagation factor: This is the coefficient used in calculating gasses' propagation by their concentration gradient. See Section 3.1 for detailed mechanism.
- Soot propagation coefficient: This is similar to the gas propagation coefficient.

All these factors have a preset range. Tuning a factor out of its range is not permitted. Also, for some of the parameters it is possible that tuning the parameter to its extreme can result in numerical instability.

5.1.2 Tuning Flooding Parameters

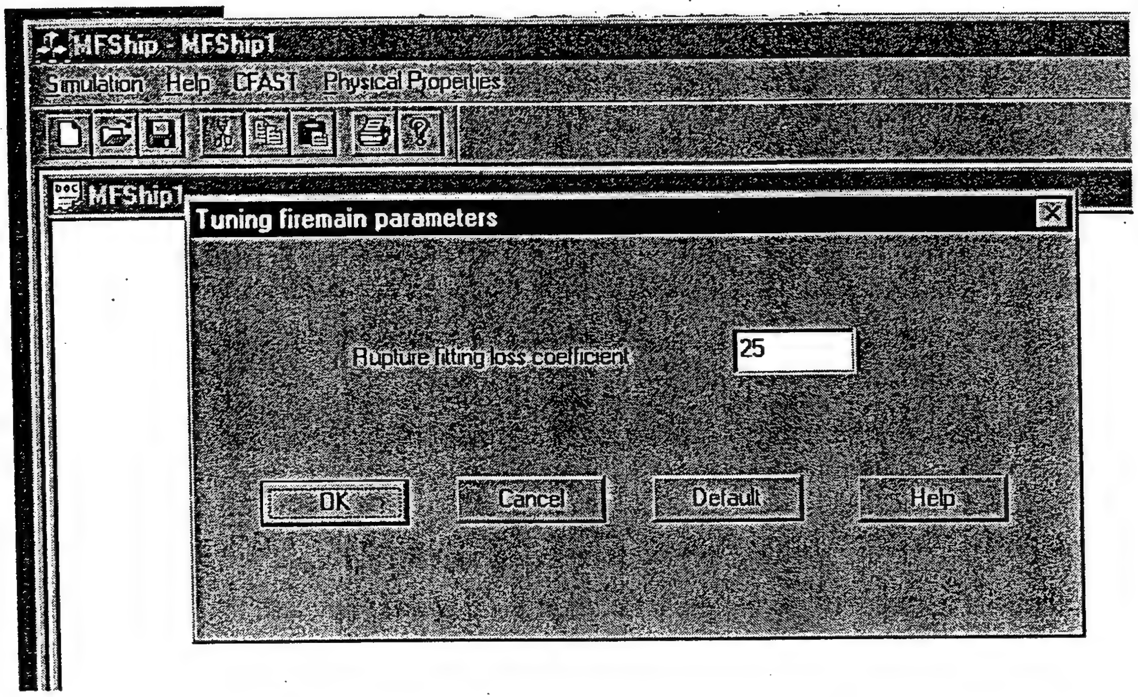

MFShip - MFShip1
Simulation Help CFAST Physical Properties
MFShip1
Tuning flooding simulation parameters
OK
Cancel
Default
Help

Flow restriction factor 0.7

Figure 7. Flooding parameter tuning dialog. This dialog is used to tune an empirical parameter used in simulating flooding

The Flooding Parameter dialog has only one parameter, the flow restriction factor. This is similar to the restriction factors for airflow discussed above in **Tuning Fire Parameters**. Increasing the flow restriction factor decreases energy loss in friction and thus speeds up water flow.

5.1.3 Tuning Fire Main Parameters



Rupture fitting loss coefficient 25

Figure 8. Fire main parameter tuning dialog

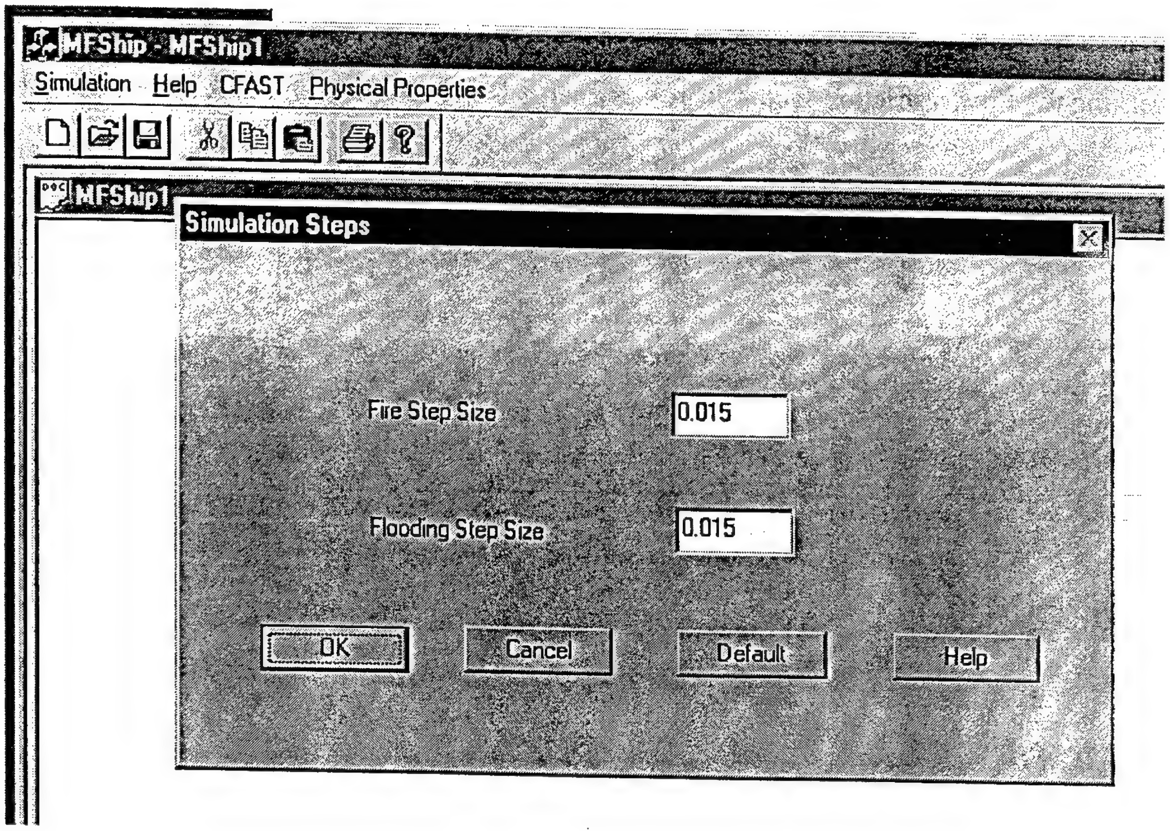
The Tuning Fire Main Parameters dialog also has only one parameter, the rupture fitting loss coefficient, which is the energy loss when water exits a pipe rupture. The fitting loss coefficient can be specified here. Increasing this factor increases energy loss and thus reduces rupture leak rate and increases fire main pressure.

5.1.4 Casualty Control Parameters

The casualty control parameters are also designed to be tunable but are not yet in place.

5.1.5 Simulation Step

The simulation step can also be tuned. Figure 9 illustrates the dialog box that shows up if the user chooses “Configure Simulation Step” from the “Simulation” menu.



Fire Step Size: 0.015

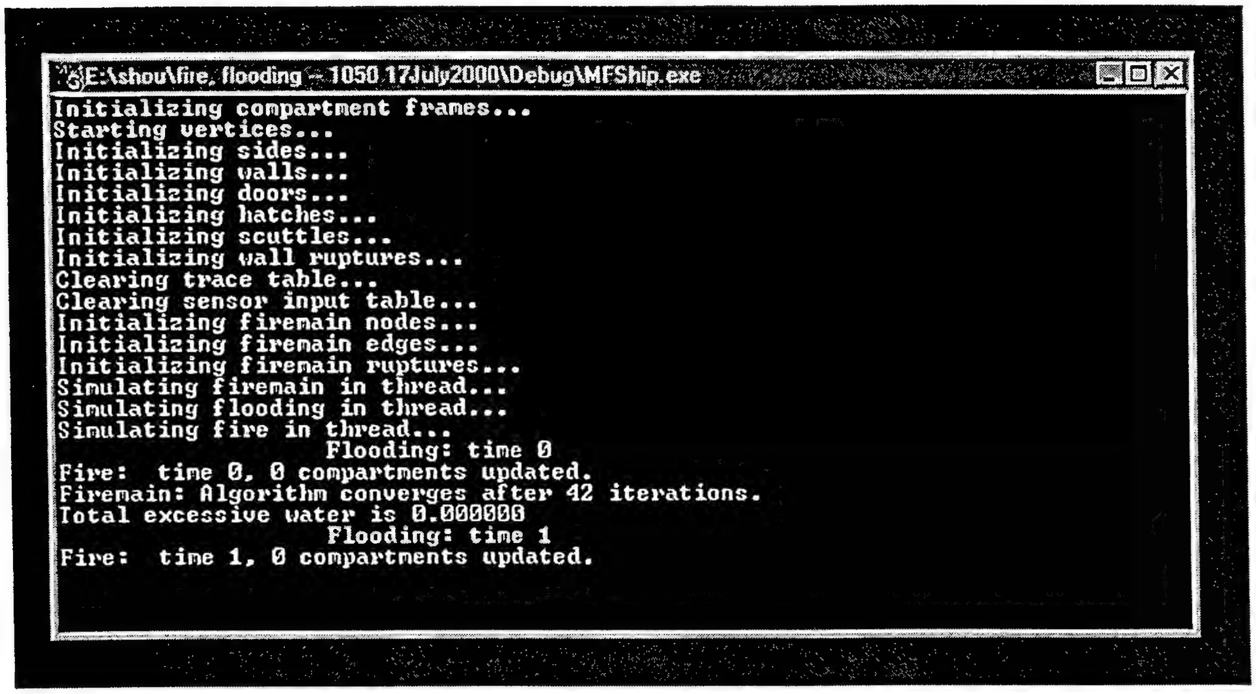
Flooding Step Size: 0.015

Figure 9. Simulation step tuning dialog

The simulation step (used in solving the ODE's) of the fire simulator and the flooding simulator can be tuned independently in this dialog. It is not recommended to use any value that is too large as this may cause large errors in numerical calculation. It is recommended to use the smallest value possible instead.

5.2 Simulation Status Monitoring

DC-SIM provides a status-monitoring window. Its main function is to print out the initialization status, current ship time, vital information for each simulator, events communicated from the ECLMessages table, and to report errors. Figure 10 shows a sample screenshot.



```
Initializing compartment frames...
Starting vertices...
Initializing sides...
Initializing walls...
Initializing doors...
Initializing hatches...
Initializing scuttles...
Initializing wall ruptures...
Clearing trace table...
Clearing sensor input table...
Initializing firemain nodes...
Initializing firemain edges...
Initializing firemain ruptures...
Simulating firemain in thread...
Simulating flooding in thread...
Simulating fire in thread...
                                Flooding: time 0
Fire:  time 0, 0 compartments updated.
Firemain: Algorithm converges after 42 iterations.
Total excessive water is 0.000000
                                Flooding: time 1
Fire:  time 1, 0 compartments updated.
```

Figure 10. Simulation status monitoring

In this sample screen shot, initialization processes are listed when they are completed. Also the fire main solver reported that its algorithm converged in 42 iterations in the first second of simulation. Flooding and fire time are also reported and they are identical. The total excessive water is the sum of the excessive water of every compartment (see Section 3.2 for how this functions), and thus provides monitoring of the effectiveness of the flooding solver.

It is planned to move this monitoring window to the MDI interface to provide scrolling and saving functions.

5.3 Auxiliary Functions

The user interface also provides auxiliary functions. One of these is CFAST access. CFAST is a validated numerical simulation for building fire, a benchmark in fire simulation area. The DC-SIM user interface provides a tool to export the ship's data into a CFAST input file format. This makes it possible to run CFAST on the data of the ship being simulated. We use this method to complete the second step of DC-SIM validation (See Section 7.2).

6. Scenario Generation

The Scenario Generation module acts as a user-friendly interface for simulation. Though during development and testing all commands are entered by hand via the central database, it is too much to ask a user to do this. It is also important to provide auxiliary features like saving, commenting and loading a scenario.

The scenario generation is currently still under development, though the majority part of the work has been done. The following is a simple introduction.

6.1 Installation

The Illinois Crisis Simulator Scenario Generator (ICSSG) consists of four files that need to be in the same directory. They are as follows:

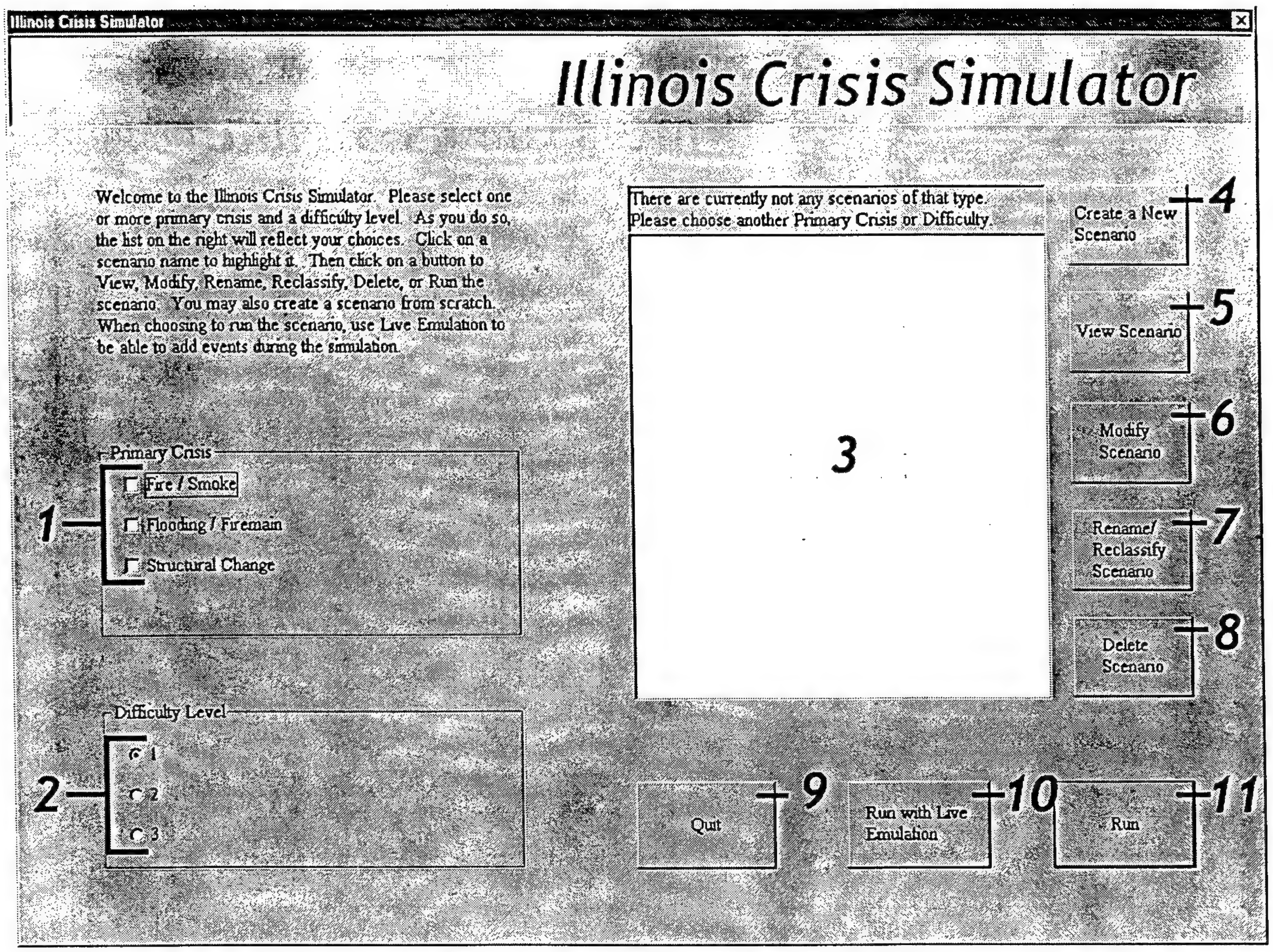
• ScenLaunch.exe	Run this program to start the Scenario Generator.
• MFShip.exe	This is the ship simulator. The Scenario Generator runs scenarios on it.
• ScenGen.dll	This is the actual program where future modifications will be made.
• SimInfo.mdb	This is the scenario information database. It stores previously created scenarios and their events.

ICSSG also needs two ODBC System DSNs configured in order to run properly. You need a DSN named *Shadwell* pointing to a ship database. This is generally a SQL Server connecting to an ARM database. The second DSN is named RINFO. It needs to point to the SimInfo.mdb in your ICSSG directory.

6.2 Interface

6.2.1 Main Menu

Upon running ScenLaunch.exe you will be greeted with the Main Menu shown in Figure 11.



Welcome to the Illinois Crisis Simulator. Please select one or more primary crisis and a difficulty level. As you do so, the list on the right will reflect your choices. Click on a scenario name to highlight it. Then click on a button to View, Modify, Rename, Reclassify, Delete, or Run the scenario. You may also create a scenario from scratch. When choosing to run the scenario, use Live Emulation to be able to add events during the simulation.

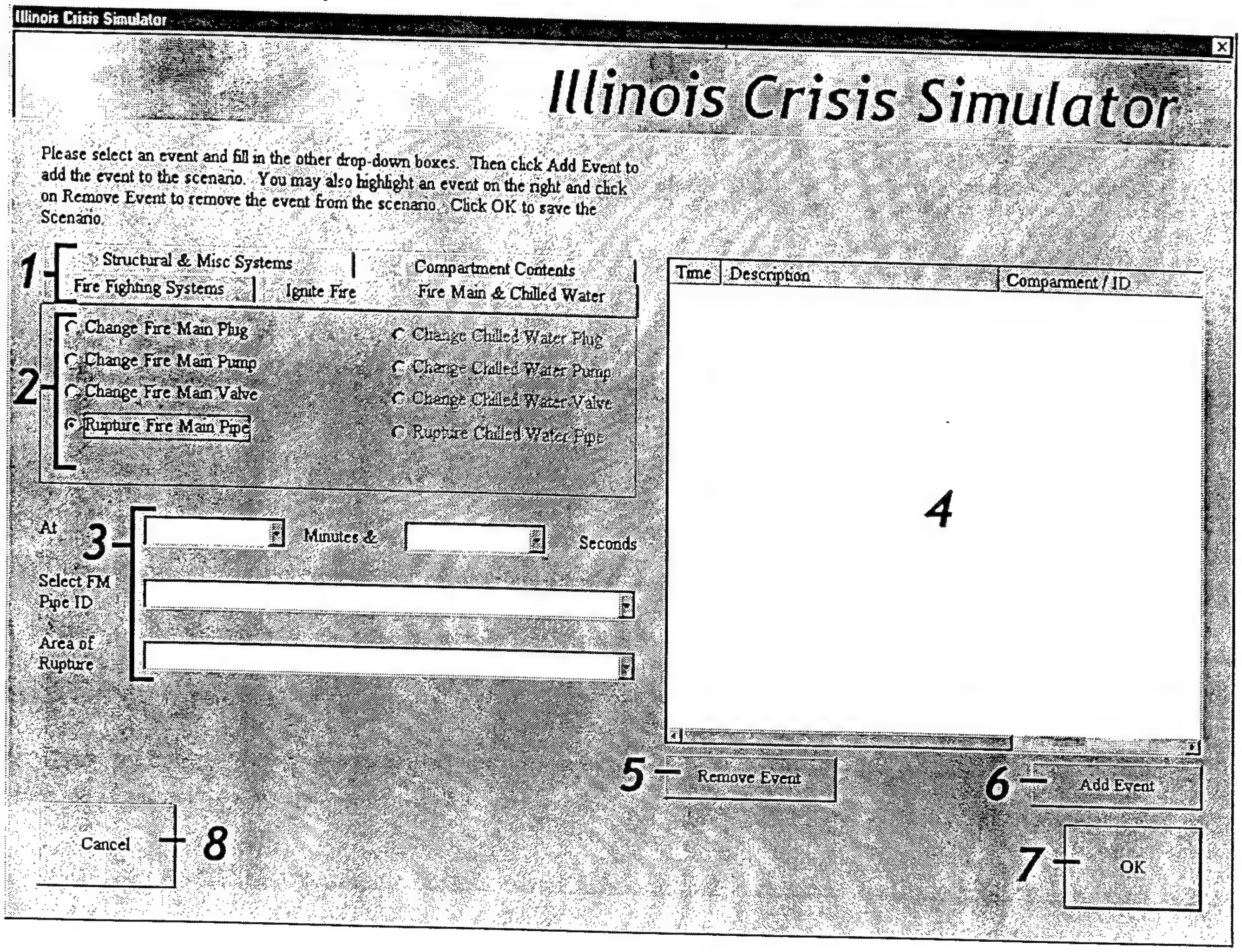
There are currently not any scenarios of that type. Please choose another Primary Crisis or Difficulty.

Figure 11. Scenario generation main menu

1. **Primary Crisis** – Select what types of events occur in the scenario.
2. **Difficulty Level** – Select the number of events in the scenario.
3. **Scenario List** – Listing of scenarios in the database that match criteria to the left.*
4. **Create a New Scenario** – Go to the scenario creation tool.
5. **View Scenario** – View the events of the currently selected scenario.
6. **Modify Scenario** – Modify the events of the currently selected scenario.
7. **Rename/Reclassify Scenario** – Change the name or the classification of the currently selected scenario.
8. **Delete Scenario** – Remove a scenario from the database permanently.
9. **Quit** – Exit ICSSG.
10. **Run with Live Emulation** – Edit a scenario while it is running (for future version).
11. **Run** – Start the simulator with the events of the currently selected scenario.

*You may also double-click any scenario listed in this window to view the contents of that scenario.

6.2.2 Create/Modify Menu



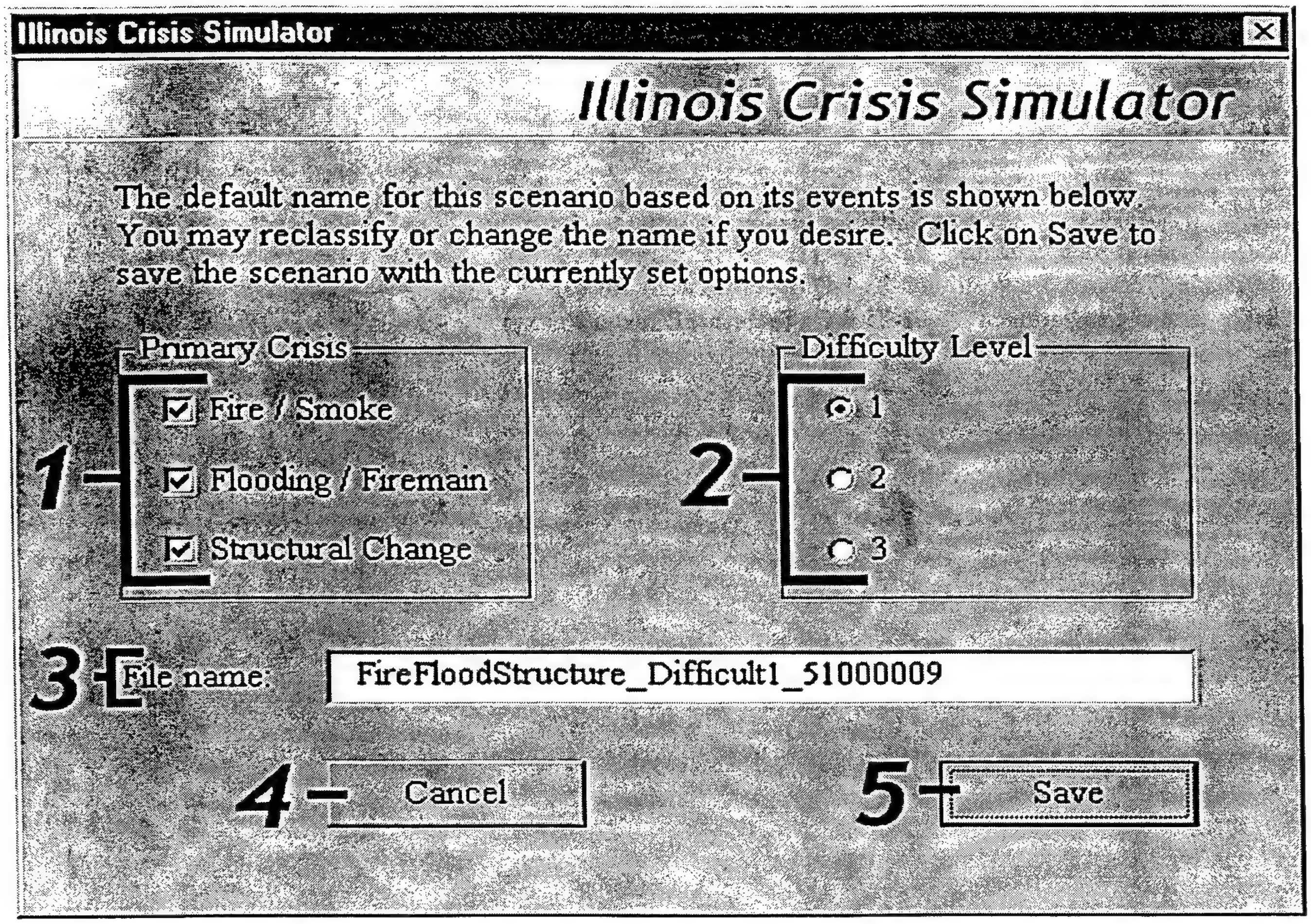
Please select an event and fill in the other drop-down boxes. Then click Add Event to add the event to the scenario. You may also highlight an event on the right and click on Remove Event to remove the event from the scenario. Click OK to save the Scenario.

Figure 12. Scenario generation create/modify menu

1. **Event Categories** – Click on a tab to list events of the selected type.
2. **Event Types** – Click on an event type to fill in information about it.
3. **Event Information** – Fill in the information for the event you desire to create.*
4. **Event Listing** – List of events in current scenario.
5. **Remove Event** – Highlight an event and click this button to remove it.
6. **Add Event** – Fill in the information on the left and click this button to add the event to the scenario.
7. **OK** – Click when finished adding/removing events to save the scenario.
8. **Cancel** – Click to cancel any changes made and to return to the main menu

* Different events need different information. Some will have dropdown boxes while others may only have 2-4 selections. You need to fill in or choose all options before you may add the event.

6.2.3 Save/Rename/Reclassify Window



The default name for this scenario based on its events is shown below. You may reclassify or change the name if you desire. Click on Save to save the scenario with the currently set options.

Primary Crisis: [✓ Fire / Smoke] [✓ Flooding / Firemain] [✓ Structural Change]

Difficulty Level: [✓ 1] 2 3

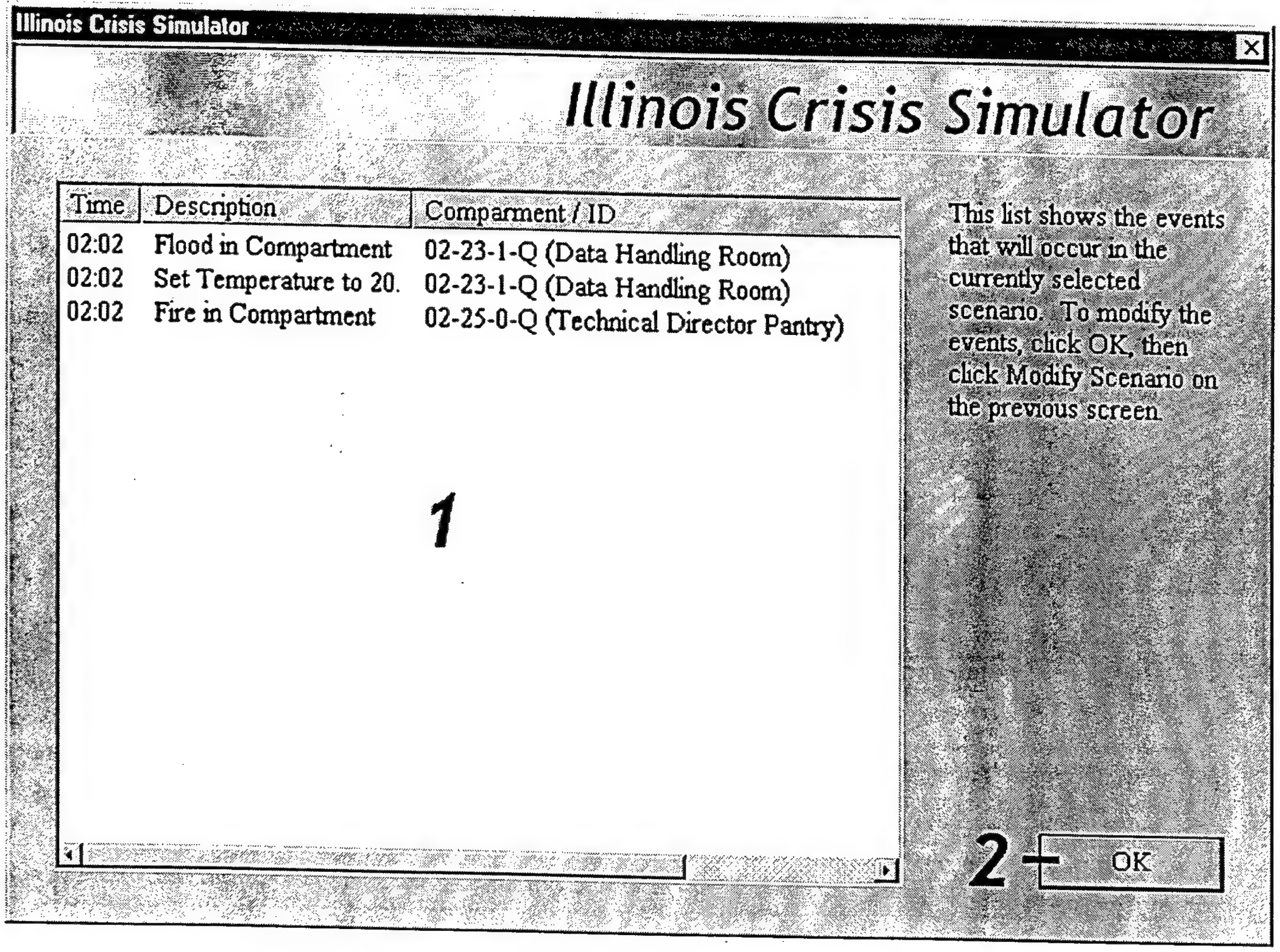
File name: FireFloodStructure_Difficult1_51000009

Figure 13. Scenario generation save/rename/reclassify window

1. **Primary Crisis** – Select what types of events occur in the scenario.*
2. **Difficulty Level** – Select the difficulty rating of the scenario.
3. **File Name** – Choose a name for the scenario.
4. **Cancel** – Click to cancel saving, renaming, or reclassifying the scenario.
5. **Save** – Click to make your changes permanent.

* The Save box will fill in classification information and a generic name based on what events are included in the scenario the first time that you save a scenario. You may change any classification information and rename the scenario if you desire.

6.2.4 View Window



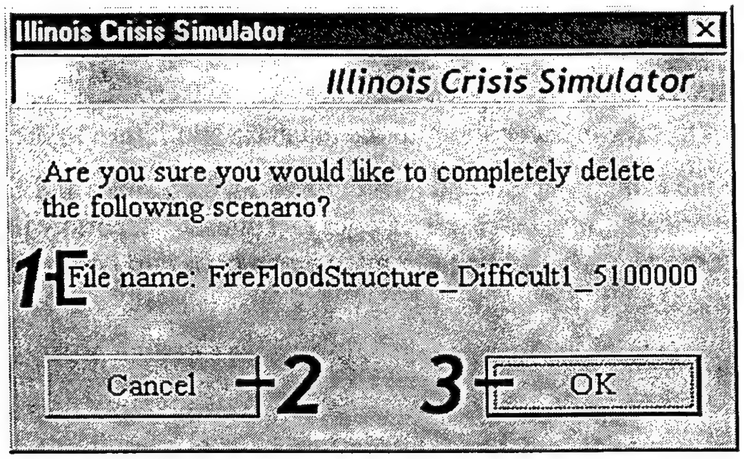
Time	Description	Comparment / ID
02:02	Flood in Compartment	02-23-1-Q (Data Handling Room)
02:02	Set Temperature to 20.	02-23-1-Q (Data Handling Room)
02:02	Fire in Compartment	02-25-0-Q (Technical Director Pantry)

This list shows the events that will occur in the currently selected scenario. To modify the events, click OK, then click Modify Scenario on the previous screen.

Figure 14. Scenario generation view window

1. **Event List** – This is a listing of the events in the scenario.
2. **OK** – Click to return to the main menu.

6.2.5 Delete Confirmation


Illinois Crisis Simulator

Illinois Crisis Simulator

Are you sure you would like to completely delete the following scenario?

File name: FireFloodStructure_Difficult1_5100000

Figure 15. Delete confirmation

1. **File Name** – This is the name of the file about to be deleted.
2. **Cancel** – Click to cancel deletion and return to the main menu.
3. **OK** – Click to permanently delete the scenario listed.

6.3 Tutorial – Creating a Scenario

The following will instruct you how to create a fire simply by increasing the temperature of a compartment.

1. Start ICSSG by running **ScenLaunch.exe**.
2. From the main menu, click on **Create a New Scenario**.
3. From the create menu, click on the **Compartment Contents** tab.
4. Click on **Change Temperature**.
5. For the time, select **0** in the minutes box and **25** in the seconds box by using the drop-down boxes.
6. Select compartment **02-23-1-Q (Data Handling Room)** from the Select Compartment box.
7. Select a temperature of **700** in the select temperature box.
8. Next, click on **Add Event**. It should show up in the Event Listings box.
9. Click on **OK**. This will bring up the save box.

10. Rename the scenario by selecting and deleting the given filename. Type **Tutorial Scenario** in the filename box.
11. Reclassify the scenario by **unchecking** the Structural Change box and **checking** the Fire/Smoke box. (Do this by clicking once on the name or the box next to the name).
12. Leave the difficulty set to **1**.
13. Click on **Save** to save the scenario into the database and return to the main menu.
14. By changing the Primary Crisis and Difficulty to **Fire / Smoke** and **1**, you will see a listing of scenarios falling under this category. Tutorial Scenario will be listed. Click on it to highlight it and click on any other main menu option to perform that option on the scenario.

6.4 Limitations

Currently ICSSG has many fields that have been grayed out and are unselectable. None of these are fully implemented by the simulator hence you may not select them. As the simulator gains functionality, so too will the scenario generator. The current list of future systems and functionality is as follows:

- AFFF fire-fighting system
- HFP fire-fighting system
- Water mist fire-fighting system
- Chilled water system
- Electrical system
- Ventilation system
- Compartment gas/smoke contents
- Compartment fuel contents

As the systems come online, you will be able to manipulate various aspects of the systems (e.g., rupture pipes or change valve status).

7. Validation of Existing Functions

This project's validation is twofold. First, we need proof of DC-SIM'S capability of accommodating DCA and DC-ARM requirements discussed in Section 1. Secondly, we want to know how well DC-SIM is as an independent ship damage control simulator.

7.1 Fulfilling DCA/ARM Requirement

DC-SIM has been used as the simulator for DC-TRAIN and DC-ARM systems. Its capability hasn't been fully revealed by the date this document is prepared. DC-TRAIN has been used at SWOS but the feedback is inadequate, and a demo for DC-ARM was scheduled but could not be completed due to funding constraints. However, based on lab testing of DC-SIM in these two projects, DC-SIM can provide reasonable fidelity and true real time response. In the next section we will provide a more rigorous comparison of DC-SIM with CFAST, a well-established fire simulator developed by the National Institute of Standards and Technology (NIST).

7.2 Comparison with CFAST Simulation

A detailed comparison is in Appendix D.

8. Extension of Existing Functions

Despite DC-SIM'S real time, environment interactive, and ship-scale simulation nature, it still provides much room for future extension. One example is to develop it into a hyper-real time predictor for crisis propagation. This is a very useful feature for AI decision-making processes used in THE DC-ARM project.

8.1 Hyper-real time prediction

As discussed before, an explicit model of the underlying physical and chemical processes can always be beneficial to AI modules. Basically the predictor can be used in this way: Upon identification of a crisis, say, a fire, the predictor can be called to guess the trend of fire propagation based on the current ship status and even counting in possible casualty control measures.

The main requirement of this extension is to provide hyper-real time performance, which means to simulate real world events at a speed hundreds of times faster. Based on the current DC-SIM'S performance on a Dell PC with dual 400MHz CPU, real time performance can be maintained when less than 300 compartments are being simulated. This is not very fast at first glance. But since the current DC-SIM interfaces other modules in DC projects every second, that occupies the majority of system time, and a hyper-real time predictor wouldn't have this requirement. Thus we are confident that the speed requirement can be reached if we further restrict the number of compartments to be simulated, i.e. to restrict the prediction locally. Since in most cases the prediction is short term, locality restriction doesn't affect prediction fidelity.

The predictor can work in this way:

Input:

- Current ship status including crisis source;
- Event sequence specifying what will happen in ascending time order;

- One or more stop criteria;

Output:

- Time and ship status when stop criteria is met;
- Trace of ship status from initial state to stop state;

The event sequence can contain possible casualty control measures taken. Several predictions can be launched at the same time with difference event sequences for comparison purpose. This is very meaningful for resource management in case multiple crises are present and only limited resources are available.

A stop criterion can be a time span (say 30 minutes), a certain event (say ignition of engine room), or both.

The implementation of this function is simple. Since DC-SIM has CShipStructure as its representation of a ship's status, the hyper-real time predictor is just a copy of the simulation engine running on a CShipStructure object without real-time timing (the multimedia timer).

9. Conclusion

Based on the design procedure and testing results, we believe that DC-SIM has fulfilled its initial design purpose, and succeeded in providing a general framework for future expansion. It also sheds light on an alternative method of simulating complex physical processes using a simplified model structure and parameter tuning. Future work could exploit the structure DC-SIM provides to see how simulation fidelity and computation speed can benefit.

10. Acknowledgments

This effort was funded in part by ONR Grant N00014-95-1-0749 and NRL Contract N00014-97-C-2061. We gratefully acknowledge the past and current KBS members who have contributed to this project. Dr. Williams Hsu, Dr. Vadim Bulitko, and Eugene Grois helped in formulating the project. Dave Kruse, Jonathan Wiswall, and Jun Zhang contributed to the simulation code. Michael Bobak, Dr. JC LeMentec and Dr. Michael Hamman coordinated the development of the simulator as a component of a larger system for ship damage control. Ron Carbonari and Peter Baer worked closely with the simulation group on development and the visualization and ontology specification. Karl Schultz and Michael Daniels provided outstanding help on database related issues. Brent Spillner assisted with executive control. Luke Stafford and Linda Chow assisted in testing the simulator. Zhijin Sheng and her hard-working system administration group maintained a highly available computer system. Misha Voloshin and Kurt Gimbel provided help with numerous issues. We also acknowledge the technical assistance in the revisions of this report by Audrey Fisher, Marcia Snow, Jamie Carras, and Nathan Otis.

11. References

Bailey, J.L., and Tatem, P.A., *Validation of Fire/Smoke Spread Model (CFAST) Using Ex-USS Shadwell Internal Ship Conflagration Control (ISCC) Fire Tests,* NRL Memorandum Report NRL/MR/6180-95-7781, September 29, 1995.

Carhart, H.W., Toomey, T.A. and Williams, F.W., "The ex-SHADWELL Full-Scale Fire Research and Test Ship" , NRL Memorandum Report NRL/MR/6180—87---6074, 6 October 1987, re-issued September 1992.

Durkin, A. F., Burne, P.C., Niciforos, P.G., Nguyen, X.H. and Williams, F.W., An Application of Process Control Tech ologies for Use Aboard Naval Ships: Automated Fire main Management System (AFMS)" NRL Ltr Rpt 6180/0181, 29 April 1998.

Gottuk, D., Hill, S.A., Schemel, C.F., Streckler, B.P., Rose-Pehrsson, S.C., Tatem, P.A. and Williams, F.W., "Identification of Fire Signatures for Shipboard Multi-Criteria Fire Detection System Analysis, NRL Memorandum Report NRL/MR/6180—99—8386, 18 June 1999.

Hill, S.A., Scheffey, J.L., Farley, J.P., Williams, F.W. *Results of the 1996 Firefighting and Damage Control Equipment Evaluation Tests (SCBA),* NRL Memorandum Report NRL/MR/6180-97-7936, March 31, 1997.

Lestina,T., Runnerstrom, E., Davis, K., Durkin, A., and Williams, F. W. (1999), *Evaluation of Firemain Architectures and Supporting Reflexive Technology, NRL Memorandum Report* NRL/MR/6180-99-8346, March 12, 1999.

Gottuk, D. and Williams, F.W., "*Multi-Criteria Fire Detection: A Review of the State-of-the-art,*" NRL Ltr Rpt 6180/0472. September 10, 1998.

Naval Sea Systems Command. (1997). *Naval Ships' Technical Manual, Chapter 555, Volume 1, Surface Ship Firefighting.* Forth Revision. S9286-S3-STM-010/CH-555V1. This chapter Supersedes Chapter 555 dated 7 March 1997.

Parker, A. J., Strehlen, B. D., Scheffey, J. L., Wong, J. T., Darwin, R. T., Pham, H., Runnerstrom, E., Lestina, T., Downs, R., Bradley, M., Toomey, T. A., Farley, J. P., Williams, F. W., Tatem, P.(1999). "Results of 1998 DC-ARM/ISFE Demonstrations Tests", NRL Ltr Rpt 6180/0032:FWW, March 12, 1999.

Portier, R. W., Reneke, P. A., Jones, W. W., and Peacock, R. D. (1992). *A User's Guide for CFAST Version 1.6.* Building and Fire Research Laboratory for the National Institute of Standards and Technology. NISTR 4985. December 1992.

Street, T. T., Bailey, J., Riddle, T. and Williams, F.W., "Upgrades to Data Handling Capabilities on ex-USS SHADWELL," NRL Ltr Rpt 6180/0229, 6 June 2000

Williams, FW., Farley, J.P, Gottuk, D.T., and Peatross, M.J. (1997). 1995 Class B Firefighting Doctrine and Tactics: Final Report. NRL Memorandum Report NRL/MR/6180-97-7909, January 13, 1997.

Williams, F.W., Farley, CDR J.P., C.W. Siegmann III, Scheffey, J.P., and Toomey, T.A. (1997). *1994 Attack Team Workshop: Phase II -- Full-Scale Offensive Fog Attack Tests,* NRL Memorandum Report NRL/MR/6180--97-7944, April 24, 1997.

Appendix A: DC-SIM Ontology

This is the specification for those parts of knowledge ontology used by DC-SIM. It is a subset of the Knowledge Ontology developed for the overall DC-SCS [Wilkins, et al, 2000b]. Additionally, this section contains illustrations related to DC-SIM.

A.1 Basic Terminology

Bow

The bow is the front of the ship.

Stern

The stern is the back of the ship.

Fore/Forward

When you're standing on the ship, fore is toward the bow. Fore is a direction, not a location. Fore is the opposite direction from aft.

Aft

When you're standing on the ship, aft is toward the stern. Aft is a direction, not a location. Aft is the opposite direction from fore.

Port

When you're standing on the ship facing the bow, port is to your left. If you're facing the stern, port is on your right. Port is a direction relative to the ship's orientation, as in "Hard [turn] to port!" Port is the opposite direction from starboard. (Having trouble remembering this? Notice that "port" and "left" both have four letters.)

Starboard

When you're standing on the ship facing the bow, starboard is to your right. If you're facing the stern, starboard is on your left. Starboard is a direction relative to the ship's orientation, as in "Hard [turn] to starboard!" Starboard is the opposite direction from port.

Waterline

Like many others, the DDG51 class of ship is expected to float on the water during normal operation. The waterline is where the top of the water meets the ship.

Global coordinate

A global coordinate system is set up on the ship for location identification. The z-axis is upward, x to bow and y to starboard. The origin is not specified by this ontology.

A.2 Static Ship Structure and Chemical Properties

This section includes tables describing a ship's basic static structure like compartments, bulkheads and vents.

A.2.1 Compartment Content

Name*	Type	Size	Notes
CompartmentID	Number (Long)	4	Foreign key on Compartments
ContentID	Number (Integer)	2	
ContentAmount	Number (Single)	4	
SuppressedAmount	Number (Single)	4	

*Used by Simulation during initialization.

ContentID is the identification of content type. Its property can be found in tables FuelDescription and ReactionProperties. ContentAmount is the amount of content, and SuppressedAmount the amount of fuel having been suppressed. Normally, when the simulator is started the SuppressedAmount is 0. But if the simulator resumes a simulation that has been interrupted or executes a simulation based on a pre-existing status, then this amount can be positive. SuppressedAmount is counted in ContentAmount.

A.2.2 Compartments

Each compartment is identified by deck, frame, position, and type. This table translates between several compartment identification schemes. "Bogus" compartments are those that do not exist on the DC plates: they are used either to enable visualization to draw elements like the ship's antenna, or to model elements such as the outside environment.

Name*	Type	Size	Notes
CompartmentID	Number (Long)	4	Serial number
Name	Text	50	Full Navy ID (i.e. 01-110-01-Q)
Deck	Number (Long)	4	KBS Deck number (i.e. –1)
Frame	Number (Long)	4	KBS Frame number (i.e. 110)
Position	Number (Long)	4	KBS Position number (i.e. 100)
Type	Text	1	Type designator (i.e. Q)
Description	Text	100	(i.e. Tech Library Annex)
KBSName	Text	50	Full KBS ID (i.e. –1-110-100)
IsBogus	Yes/No	1	Marks "fake" compartments

*Used by Simulation during initialization.

The simulation doesn't use position information for compartments. For simulation, the only significant information provided by this table is the ID and name of the compartments.

A.2.3 Doors

This table characterizes the doors on the ship with regard to their position within the containing bulkhead. Doors with type –1 are internal to a compartment; other doors connect compartments.

Name*	Type	Size	Notes
DoorID	Number (Long)	4	Serial number
WallID	Number (Integer)	2	Foreign key on Walls
CenterX	Number (Integer)	2	
CenterY	Number (Integer)	2	
CenterZ	Number (Integer)	2	
Width	Number (Single)	4	[ft]
Height	Number (Single)	4	[ft]
Name	Text	50	Canonical Navy ID
Type	Number (Long)	4	(see DC Plate 3)

*Used by Simulation during initialization.

Doors can be seen as yellow spheres in the ship visualization.

WallID is the identification of the bulkhead that contains the door. CenterX, CenterY and CenterZ are the coordinates of the door's center based on the global coordinate system. Width and Height describe the door's dimensions.

A.2.4 Fire Main Nodes

The fire main is modeled as an undirected graph; this table describes the nodes of the fire main. A node is anything that can affect flow within the fire main system: valves, junctions, narrowings, widenings, gauges, and pumps, for instance. The handle fields are used to describe certain kinds of connection info, as any given node may only be the endpoint of at most three edges. So an end node, like a sprinkler, will only connect to one edge; this edgeID is stored in HandleOne. A node connecting two edges (such as an elbow joint) will have values for HandleOne and HandleTwo. For a node incident to three edges (such as a T-junction), all three handles will contain the ID's of the incident edges. Any unused handles will contain –1 as their value.

Name*	Type	Size	Notes
NodeID	Number (Long)	4	Serial number
X	Number (Single)	4	
Y	Number (Single)	4	
Z	Number (Single)	4	
HandleOne	Number (Long)	4	Foreign key on FireMainPipes
HandleTwo	Number (Long)	4	Foreign key on FireMainPipes
HandleThree	Number (Long)	4	Foreign key on FireMainPipes
InSea	Yes/No	1	
CompartmentID	Number (Long)	4	Foreign key on Compartments
VertexType	Text	50	Describes the nature of node
Name	Text	50	Navy ID (valves only)

*Used by Simulation during initialization.

X, Y and Z are the coordinates of the node. Handles indicate the incident edges (pipe or pump). There can be up to three incident edges. The check box InSea indicates if the node is immersed in water or not. Seawater intake is surely in sea. But an open node in a flooded compartment can also be in sea. CompartmentID identifies the compartment where the node is located.

The NodeIDs form a continuous range, including valves, fire plugs, and pressure sensors. Specific nodes like valves appear in individual tables, but they are also included in this table.

A.2.5 FireMainPipes

This table lists the edges of the fire main graph.

Name*	Type	Size	Notes
PipeID	Number (Long)	4	Serial number
HeadNodeID	Number (Long)	4	Foreign key on FireMainNodes
TailNodeID	Number (Long)	4	Foreign key on FireMainNodes
CompartmentID	Number (Long)	4	Foreign key on Compartments
Diameter	Number (Single)	4	[ft]

*Used by Simulation during initialization.

A pipe is directional: from tail node to head node. Pipes, together with the fire main pumps, form the edges of the fire main directional graph.

A.2.6 FuelDescription

This table is used to hold important information about the different types of fuel that would be present on the ship.

Name*	Type	Size	Notes
Identifier	Number (Integer)	2	Foreign key on ReactionProperties
FuelName	Text	50	
UnitFuelConsumingRate	Number (Double)	8	Kg/s
CompleteFuelHeatCapacity	Number (Double)	8	J/kg
IncompleteFuelHeatCapacity	Number (Double)	8	J/kg
FuelBasicSootCapacity	Number (Double)	8	Kg
FuelIncompleteSootCapacity	Number (Double)	8	Kg
MolarMass	Number (Double)	8	Kg

*Used by Simulation during initialization.

This is not really part of the ship structure, but is instead the fuel's chemical properties. Identifier is the ID used in table CompartmentContent and ReactionProperties. UnitFuelConsumingRate is the normal consumption rate of a unit (1 kg) of fuel if it is ignited. This is a rough estimation of combustion since strictly the speed is not a constant. CompleteFuelHeatCapacity indicates how much heat is released by burning one unit of fuel under complete combustion, and IncompleteFuelHeartCapacity how much is released under incomplete combustion. FuelBasicSootCapacity indicates how much soot is released by burning one unit of fuel under complete combustion, and FuelIncompleteSootCapacity how much is released under incomplete combustion.

A.2.7 Hatches

Hatches are openings in decks and overheads; this table is otherwise identical to the Doors table.

Name*	Type	Size	Notes
HatchID	Number (Long)	4	Serial number
WallID	Number (Integer)	2	Foreign key on Walls
CenterX	Number (Integer)	2	
CenterY	Number (Integer)	2	
CenterZ	Number (Integer)	2	
HatchWidth	Number (Single)	4	[ft]
HatchHeight	Number (Single)	4	[ft]
Name	Text	50	Canonical Navy ID
Type	Number (Long)	4	(see DC Plate 3)

*Used by Simulation during initialization.

WallID is the identification of the deck that contains the hatch. CenterX, CenterY and CenterZ are the coordinates of the hatch's center based on the global coordinate system. Width and Height describe the hatch's dimensions.

A.2.8 ReactionProperties

This Table is used to describe the chemical properties of certain reactions, and is used in conjunction with the FuelDescription table to accurately simulate anything involving fuel.

Name*	Type	Size	Notes
ReactionID	Number (Long)	4	
Description	Text	50	
ReactionType	Text	50	
O_2	Number (Single)	4	
CO_2	Number (Single)	4	
CO	Number (Single)	4	
H_2O	Number (Single)	4	
SO_2	Number (Single)	4	
SOOT	Number (Single)	4	
HEAT	Number (Single)	4	
Key	Number (Long)	4	Serial number

*Used by Simulation during initialization.

Like FuelDescription, this table describes chemical properties. A record of the table actually describes a chemical reaction equation. ReactionID is the ID of the combustible (the same as ContentID in CompartmentContent table and Identifier in FuelDescription table). The values of O_2, CO_2, CO, H_2O and SO_2 indicate the coefficient of the reaction equation: negative for reactant and positive for product. The coefficients are normalized by letting the combustible's coefficient be 1. SOOT and HEAT are similar to those described in table FuelDescription, and are actually not used by the simulation.

A.2.9 Scuttles

Scuttles are circular openings within hatches.

Name*	Type	Size	Notes
ScuttleID	Number (Long)	4	Serial number
WallID	Number (Long)	4	Foreign key on Walls
Name	Text	50	Navy name for scuttle
CenterX	Number (Single)	4	
CenterY	Number (Single)	4	
CenterZ	Number (Single)	4	
Radius	Number (Single)	4	[ft]
HatchID	Number (Long)	4	Foreign key on Hatches

*Used by Simulation during initialization.

WallID identifies the deck that contains the scuttle. CenterX, CenterY and CenterZ are the coordinates of the scuttle's center based on the global coordinate system. Radius is the scuttle's radius. HatchID identifies the hatch that contains the scuttle.

A.2.10 Sides

A side (one face of a bulkhead) is made of four vertices, listed in counterclockwise order with respect to their parent compartment. In particular, with a vertical side, vertices 2 and 3 will be the top vertices. Sides of type 0 are overheads, type 1 are decks and type 2 are vertical sides.

Name*	Type	Size	Notes
SideID	Number (Long)	4	Serial number
VertexID1	Number (Long)	4	Foreign key on Vertices
VertexID2	Number (Long)	4	Foreign key on Vertices
VertexID3	Number (Long)	4	Foreign key on Vertices
VertexID4	Number (Long)	4	Foreign key on Vertices
SideType	Number (Long)	4	Ceiling, floor, side
Level	Number (Long)	4	Level number
CompartmentID	Number (Long)	4	Foreign key on Compartments

*Used by Simulation during initialization.

A.2.11 Vertices

Lists all of the vertices in the ship.

Name*	Type	Size	Notes
VertexID	Number (Long)	4	Serial number
X	Number (Single)	4	
Y	Number (Single)	4	
Z	Number (Single)	4	

*Used by Simulation during initialization.

A.2.12 Bulkheads

Each bulkhead is composed of two sides. In some cases, the same compartment is on both sides of the bulkhead; these bulkheads are visualization artifacts. For horizontal bulkheads, the first side is always the overhead and the second is the deck.

Name*	Type	Size	Notes
WallID	Number (Long)	4	Serial number
SideID1	Number (Long)	4	Foreign key on Sides
SideID2	Number (Long)	4	Foreign key on Sides
CompartmentID1	Number (Long)	4	Foreign key on Compartments
CompartmentID2	Number (Long)	4	Foreign key on Compartments
Thickness	Number (Single)	4	[units]
Material	Text	50	
WallType	Number (Long)	4	
OpeningType	Number (Long)	4	

*Used by Simulation during initialization.

SideID1 is the ID of side 1 which is in the compartment with ID CompармtentID1. Similarly SideID2 is the ID of side 2 for CompartmentID2.

A.3 Dynamic Ship Status

This section includes tables describing aspects of ship status that can change.

A.3.1 CompartmentStatus

Lists the current physical status of each compartment.

Name*	Type	Size	Notes
CompartmentID	Number (Long)	4	Foreign key on Compartments
LowerZoneHeight	Number (Single)	4	[feet]
UpperZoneHeight	Number (Single)	4	[feet]
LowerZoneTemperature	Number (Single)	4	[Kelvin]
UpperZoneTemperature	Number (Single)	4	[Kelvin]
Pressure	Number (Single)	4	[pascals]
O2Concentration	Number (Single)	4	[mol/m^3]
FireStatus	Text	50	Ignited, engulfed, destroyed, extinguished, intact
FuelAmount	Number (Single)	4	[kg]
SootDensity	Number (Single)	4	[kg/m^3]
WaterDepth	Number (Single)	4	[meters]
FloodingStatus	Text	50	Intact, flooded
CO2Concentration	Number (Single)	4	[mol/m^3]
COConcentration	Number (Single)	4	[mol/m^3]
HFConcentration	Number (Single)	4	[mol/m^3]
HCLConcentration	Number (Single)	4	[mol/m^3]
HBrConcentration	Number (Single)	4	[mol/m^3]
CombustibleFuelAmount	Number (Single)	4	[kg]

*Used once per second by Simulation to report ship status.

A.3.2 CompartmentStatusTrace

Tracks "important" changes to the CompartmentStatus table. For notes on the column values, see the CompartmentStatus table. Note that a '-2' value for CompartmentID marks a special 'parameters entry'.

Name*	Type	Size	Notes
TraceID	Number (Long)	4	Serial number
TimeTag	Number (Long)	4	
CompartmentID	Number (Long)	4	Foreign key on Compartments
LowerZoneHeight	Number (Single)	4	
LowerZoneTemperature	Number (Single)	4	
UpperZoneTemperature	Number (Single)	4	
Pressure	Number (Single)	4	
FireStatus	Text	50	
FloodingDepth	Number (Single)	4	[meters]
SmokeDensity	Number (Single)	4	

*Used by Simulation to record simulation history.

The unit of SmokeDensity can be different from that of SootDensity in the CompartmentStatus table.

A.3.3 DoorStatus

Lists whether each door on the ship is open or closed.

Name*	Type	Size	Notes
DoorID	Number (Long)	4	Foreign key on Doors
IsOpen	Yes/No	1	

*Used by Simulation during status changes.

A.3.4 FireBoundaries

This table describes fire boundaries set by damage control.

Name*	Type	Size	Notes
Set	Number (Long)	4	Set # this belongs to
SideID	Number (Long)	4	Foreign key on sides

*Used by Simulation during status changes.

Fire boundaries are grouped into sets. A set contains several bulkheads on which a fire boundary should be set.

A.3.5 FireMainHoses

Lists additional hose added to the fire main.

Name*	Type	Size	Notes
FirePlugID	Number (Long)	4	Foreign key on FM pipes or pumps
Diameter	Number (Single)	4	[foot]
Length	Number(Single)	4	[foot]
DestCompartmentID	Number(Long)	4	Foreign key on Compartments
Removed	Yes/No	1	Removed or not

*Simulation uses this table to gain information on adding/removing a hose.

A hose always originates at a fireplug, and ends at some compartment to which it directs water.

A.3.6 FireMainFlowRates

Shows current flow rates in the fire main (FM).

Name*	Type	Size	Notes
EdgeID	Number (Long)	4	Foreign key on FM pipes or pumps
FlowRate	Number (Single)	4	[gal/minute]

*Used by Simulation once per FM solving → per FM structure change.

A.3.7 FireMainPlugs

Lists the status of plugs in the fire main system.

Name*	Type	Size	Notes
PlugID	Number (Long)	4	Foreign key on FireMain(New)Nodes
IsOpen	Yes/No	1	

*Used by Simulation during initialization and change of plug status.

A.3.8 FireMainPressures

Lists the pressure at each fire main node.

Name*	Type	Size	Notes
NodeID	Number (Long)	4	Foreign key on FireMain(New)Nodes
Handle	Number (Long)	4	Foreign key on FireMain(New)Pipes
Pressure	Number (Single)	4	[psi]

*Used by Simulation during status changes (same as FireMainFlowRates).

A.3.9 MainPumps

Lists the status of fire main pumps.

Name*	Type	Size	Notes
PumpID	Number (Long)	4	Foreign key on FireMain(New)Pipes
MaximumFlowRate	Number (Single)	4	[gal/minute]
MaximumHeadGain	Number (Single)	4	[psi]
TailNodeID	Number (Long)	4	
HeadNodeID	Number (Long)	4	
Status	Text	50	On, off, standby, overheating, damaged, destroyed.
PowerLevel	Number (Single)	4	[percentage]
CompartmentID	Number (Long)	4	Foreign key on Compartments
IsOn	Yes/No	1	
Name	Text	50	Pump number

*Used by Simulation during status changes and initialization.

The pump's characteristics are modeled as a quadratic curve with parameter MaximumFlowRate and MaximumHeadGain.

A.3.10 FireMainRuptures

Lists fire main ruptures formed during simulation.

Name*	Type	Size	Notes
RuptureID	Number (Long)	4	Serial number
PipeID	Number (Long)	4	Foreign key on FireMain(New)Pipes
Position	Number (Single)	4	Offset from pipe tail [ft]
CrackWidth	Number (Single)	4	[ft]
CrackLength	Number (Single)	4	[ft]
LeakRate	Number (Single)	4	[gal/s]
Patched	Yes/No	1	Has it been patched or not

*Used by Simulation during status changes and initialization.

The rupture is modeled as an open mouth. CrackWidth is the maximum distance between the two lips, and CrackLength is the perimeter of the opening. Upon solving the fire main, the leak rate is reported here. RuptureID is an auto number. In order to specify a rupture, this ID should be used instead of PipeID. By doing this we disallow setting more than one rupture on a single pipe.

A.3.11 FireMainValves

Lists the current status of each valve on the ship.

Name*	Type	Size	Notes
ValveID	Number (Long)	4	Node identifier
IsOpen	Yes/No	1	
IsRemoteControllable	Yes/No	1	
Status	Text	50	Navy status code

*Used by Simulation during status changes and initialization.

A.3.12 HatchStatus

Lists whether each hatch on the ship is open or closed.

Name*	Type	Size	Notes
HatchID	Number (Long)	4	Foreign key on Hatches
IsOpen	Yes/No	1	

*Used by Simulation during status changes and initialization.

A.3.13 ScuttleStatus

Lists whether each scuttle on the ship is open or closed.

Name*	Type	Size	Notes
ScuttleID	Number (Long)	4	Foreign key on Scuttles
IsOpen	Yes/No	1	

*Used by Simulation during status changes and initialization (same as HatchStatus).

A.3.14 WallRuptures

Lists bulkhead ruptures formed during simulation.

Name*	Type	Size	Notes
RuptureID	Number (Long)	4	Serial number
WallID	Number (Long)	4	Foreign key on Walls
Height	Number (Single)	4	[m] from deck to center of rupture
RuptureArea	Number (Single)	4	[ft^2]
RuptureClass	Text	50	

*Used by Simulation during status changes and initialization (once per rupture added).

RuptureID is the one use to specify a rupture via ECL. So it is possible to set more than one rupture on a single bulkhead.

A.4 DC-ARM Specific Tables

Some tables are DC-ARM specific. They are included in this section. Based on the different structure of Ontology, DC-SIM also has two different versions, to deal with DC-ARM specific features.

A.4.1 CompartmentSensors - Static

This table lists for each sensor its location, its channel number, and the compartment it is in.

Name	Type	Size	Notes
SensorID	Number (Long)	4	Serial number
SensorType	Text	50	
X	Number (Single)	4	
Y	Number (Single)	4	
Z	Number (Single)	4	
Channel#	Text	50	
Location	Text	50	Canonical name
Compartment	Number (Long)	4	Foreign key on compartments

This table is used by Simulation if sensor output is enabled.

X,Y, and Z are the coordinate of the sensor. Channel# is the MASSCOMP channel number for the sensor. SensorType indicates what kind of sensor it is. This can be, for instance, UpperZoneTemperature, O_2 or Obscuration.

A.4.1 FireMainSensors - Static

This table lists the fire main sensors and their locations.

Name	Type	Size	Notes
SensorID	Number (Long)	4	Serial number
SensorType	Text	50	Abbreviation of type of sensor
X	Number (Long)	4	
Y	Number (Long)	4	
Z	Number (Long)	4	
Channel#	Text	50	
Location	Number (Long)	4	Canonical location on ship
CompartmentID	Number (Long)	4	Foreign key on compartments
PositionID	Number(Long)	4	Foreign key on PipeID, NodeID, or PumpID

This is the same as CompartmentSensors except that the additional field PositionID indicates the position of the sensor on fire main network. For a pressure sensor, the position ID is a node ID, and for a flow rate sensor, it is a pipe ID or pump ID.

A.4.2 SensorInput - Dynamic

This table is used to track the different values the sensors give over the course of a simulation.

Name	Type	Size	Notes
Serial	Number (Long)	4	Serial number
TimeStamp	Number (Long)	4	Time the sensor is giving information
CompartmentID	Number (Long)	4	Foreign key on Compartments
SensorType	Text	50	
Value	Number (Single)	4	Information from sensor

When sensor output is enabled, the simulation reports sensor readings to this table.

A.5 ECL Communication Language

The tables in this part deal with ECL communication.

A.5.1 ECL Grammar

A machine-readable grammar for ECL (Event Communication Language) messages.

Name	Type	Size	Notes
Number	Number (Long)	4	Message number
Message	Memo	-	Description
Parameters	Memo	-	Argument
Example	Memo	-	Example text
Grammar	Memo	-	Composition
ParameterExample	Memo	-	Parameter desc.

This table just explains each ECL command.

A.5.2 ECL Messages

All communications between modules occurs in this table.

Name	Type	Size	Notes
Serial	Number (Long)	4	Serial number
ECLNum	Number (Long)	4	Message number
Timestamp	Number (Long)	4	Timestamp
To	Text	50	Addressee
From	Text	50	Sender
Problem	Text	50	Parameter 1
System	Text	50	Parameter 2
Alarm	Text	50	Parameter 3
Status	Text	50	Parameter 4
Adjective	Text	50	Parameter 5
Compartment	Text	50	Parameter 6
String1	Text	50	Parameter 7
String2	Text	50	Parameter 8
Num1	Number (Long)	4	Parameter 9
Num2	Number (Long)	4	Parameter 10
Saft	Number (Long)	4	Parameter 11
Paft	Number (Long)	4	Parameter 12
Pfor	Number (Long)	4	Parameter 13
Sfor	Number (Long)	4	Parameter 14
Above	Number (Long)	4	Parameter 15
Below	Number (Long)	4	Parameter 16

Simulation receives commands on ship status change via this table exclusively. For detailed usage, refer to Appendix B.

A.5.3 ECL Language Usage:

This is part of the ECL Grammar table. We enclosed it here for quick reference.

9101: Minutes since General Quarters (GQ)
[num1] is number of minutes since GQ set

9102: Seconds since GQ
[num1] is number of seconds since GQ set

9105: New time scaling factor
[num1] is current simulation time in seconds; [num2] is new time scaling factor * 100

9110: Begin simulation

9111: Pause simulation

9112: Resume simulation

9113: Stop simulation

9120: Call GQ. Sets zero point for simulation time.

9201: Dynamic table updated by agents
[string1] is table name; [num1] is serial number of updated entry

9202: Dynamic table updated by simulator
[string1] is table name; [num1] is serial number of updated entry

9211: Fighting fire
[num1] is compartment serial number; [string1] is fire-fighting medium; [num2] represents strength (number of 'doses') of fighting

9212: Flooding compartment
[num1] is compartment serial number

9213: Dewatering compartment
[num1] is compartment serial number

9214: Mechanically isolating compartment
[num1] is compartment serial number

9215: Electrically isolating compartment
[num1] is compartment serial number

9216: Overhauling compartment
[num1] is compartment serial number

9231: Ventilate (desmoke) compartment
[num1] is compartment serial number

9232: Secure (stop) ventilation in compartment
[num1] is compartment serial number

10001: Ignite compartment (Temporary hack for primary damage)
[num1] is compartment serial number

10002: Flood compartment (Temporary hack for primary damage)
[num1] is compartment serial number

12821: "God mode" status report from sim
[compartment] is KBS compartment ID; [status] is new status

Appendix B: Examples

B.1 Using Various Simulation Features

The following is a list of how to use special simulation features. The numbers (ship part ID, and time in seconds) appearing in the examples are chosen arbitrarily. If you are going to use that feature on a specific ship database, you need to make sure that the numbers you do use make sense on your ship.

B.1.1 Ignite a Compartment

There are two ways to ignite a compartment. One is to use the God-mode command ECL 10001, and the other is to specify the compartment's status in such a way that its temperature is high enough and oxygen concentration is not too low. Here we explain the ECL10001 method. The following ECL message ignites compartment 128 at ship time 109 (time in seconds).

ECLNum = 10001, TimeStamp = 109, Num1 = 128

B.1.2 Specify Compartment Initial Status

A compartment's status can be specified intentionally to give the simulator an initial state to run. The following procedure specifies the state of compartment 128 at ship time 130 and then notifies the change to the simulator.

Step1: Edit CompartmentStatus table so that the record with CompartmentID 128 has:

LowerZoneTemperature = 300(Celsius) , UpperZoneHeight = 300(Celsius). It is **not recommended** to change the Pressure field or to assign a temperature value that is too large.

Step2: Send the following ECL command:

ECLNum = 9201, TimeStamp = 130, String1 = "CompartmentStatus", ECLNum1 = 128.

After simulator reads in the ECL command, it retrieves the record with CompartmentId = 128 in the CompartmentStatus table, and then 300 degree (Celsius) automatically ignites the compartment.

What can be specified as initial state are Temperatures, Pressure and O_2Concentration.

B.1.3 Open/Close a Door/Hatch/Scuttle

Opening or closing a door, a hatch or a scuttle are very similar to each other. Here we give an example of opening door 80 at time 239:

Step1: In DoorStatus table, check the IsOpen box for door 80.

Step2: Send the following ECL command to ECLMessages table:

ECLNum = 9201, TimeStamp = 239, String1 = "DoorStatus", Num1 = 80.

B.1.4 Add/Patch a Bulkhead Rupture

The following is an example of adding a rupture on bulkhead 35 at time 345:

Step1: Add the following information to WallRuptures table:

> WallID = 35, Area = 9.0, Height = 1.0. Area is in square feet. Height is the height of the rupture's center on the wall if the wall is vertical. If it is a deck, then field Height has no significance. Field class has no significance.

Step2: Send the following ECL command:

> ECLNum = 9201, TimeStamp = 345, String1 = "WallRuptures", Num1 = the rupture's ID that is specified by database automatically after the bulkhead rupture record is inserted in the 1st step.
>
> Note: Patching a bulkhead has not been implemented..

B.1.5 Set Fire Boundary

The following 2 steps show setting up a set of fire boundaries on side 34, 37 and 39 with set ID 2 at time 501.

Step1: In FireBoundaries table, add three records:

> Set = 2, SideID = 34; Set = 2, SideID = 37; Set = 2, SideID = 39.

Step2: Send the following ECL command:

> ECLNum = 9221, TimeStamp = 501, Num1 = 2 (set ID).

Upon retrieving this command, the simulation queries the ProposeFireBoundaries table and sets a fire boundary on all the sides with Set ID 2.

B.1.6 Fight Fire Using Water/Water Mist/AFFF/Halon

Sending the following ECL command fights fire using water mist at compartment 128 at time 567 with strength 95%:

> ECLNum = 9211, TimeStamp = 567, String1 = "watermist", Num1 = 128, Num2 = 95.

Num2 must be an integer in range 0 ~ 100. For fighting fire using other methods, specify String as "water", "afff" or "halon" instead.

B.1.7 Flood a Compartment

The following God-mode ECL command initiates flooding of compartment 127 at time 234:

> ECLNum = 10002, TimeStamp = 234, Num1 = 127.

There are other (more natural) ways to initiate flooding of a compartment. One is to add a bulkhead rupture of compartment 127. Make it sure that the bulkhead with rupture is facing the ocean and the rupture is beneath the water surface. Water surface height is specified in table CompartmentStatus. It is the WaterDepth of the record with CompartmentID 0. This information has to be in the database before running the simulation.

The second way to flood a compartment is to add a fire main rupture at a convenient pipe segment. This will be discussed later.

B.1.8 De-flood a Compartment

The following ECL command initiates de-flooding of compartment 93:

> ECLNum = 9213, TimeStamp = when you start, Num1 = 93, Num2 = the deflooding rate you can specify (in cubic meter per second). Num2 can be unspecified (0) and in this case the simulator will use the default rate.

The capability to stop de-flooding has not been implemented.

Open/close a fire main valve

The following two steps opens valve 23 at time 45:

Step1: Check the IsOpen box of valve 23 in table FireMainValves.

Step2: Send the following ECL command:

> ECLNum = 9201, TimeStamp = 23, String1 = "FireMainValves", Num1 = 23.

To close a valve, just repeat the above process by unchecking the IsOpen box.

B.1.9 Open/Close a Fire Main Sprinkler/Plug

The following two steps opens plug 23 at time 45:

Step1: Check the IsOpen box of plug 23 in table FireMainPlugs.

Step2: Send the following ECL command:

> ECLNum = 9201, TimeStamp = 23, String1 = "FireMainPlugs", Num1 = 23.

To close the plug just repeat the above procedure by unchecking the IsOpen box.

B.1.10 Turn on/off a fire main pump

The following turns on pump 415 at time 54:

Step1: Check box IsOn of record with PumpID 415 in FireMainPumps table.

Step2: Send the following ECL command:

> ECLNum = 9201, TimeStamp = 54, String1 = "FireMainPumps", Num1 = 415.

To turn off a pump just uncheck the IsOn box and send the ECL command.

B.1.11 Add/Patch a Fire Main Pipe Rupture

The following 2 steps add a rupture to a pipe segment.

Step1: Add a new record to FireMainRuptures table: PipeID=21, Position = 0.2(feet from head of pipe 21), CrackWidth = 0.05 (foot), CrackLength = 0.4(foot), Patched unchecked.

Comments:

1. Do not assign Position a value greater than the pipe's length, or else an assertion failure will occur.
2. A rupture is modeled as a crack in the pipe. The crack is assumed to be like a half-open mouth. The crack width is the distance between the middle point of the upper lip and lower lip. The crack length is the total length of the crack perimeter.

Step2: Send the following ECL command:

ECLNum = 9201, TimeStamp = what you want, String1 = "FireMainRuptures", Num1 = the RuptureID created automatically. (This means you need to read in that record again to get the RuptureID after you put it into the FireMainRuptures table.)

To patch a rupture, just check the Patched box, and send an ECL command in the same format.

B.1.12 Add a Hose to Direct Water From a Remote Site

There are three points to specify: the fireplug where water is from, the diameter of the hose, the length of the hose, and the compartment to which water is directed. All these pieces of information are fields in the FireMainHoses table.

Step1: Add a record describing the hose to the table FireMainHoses:

FirePlugID = 42, Diameter = 0.3 (foot), Length = 20 (feet), DestCompartmentID = 111, box "Removed" unchecked.

Step2: Send ECL command:

ECLNum = 9201, TimeStamp = when the hose is added, String1 = "FireMainHoses", Num1 = 42 (the plug's ID).

To remove a hose just check the box "Removed" and send the same ECL command.

Important: If a hose is connected to fire plug 42 which has incident pipe 41, then pipe 41 cannot have any rupture. If it has, then the rupture leak will be incorrectly added to compartment 111 instead of compartment 103 where the pipe is. To add a rupture, add it to a nearby pipe.

B.2 An Example of DC-SIM Running on a Complicated Scenario

In this section we give an example of DC-SIM duplicating scenario No. 7 of the 1998 DC-ARM/ISFE Demonstration Tests.

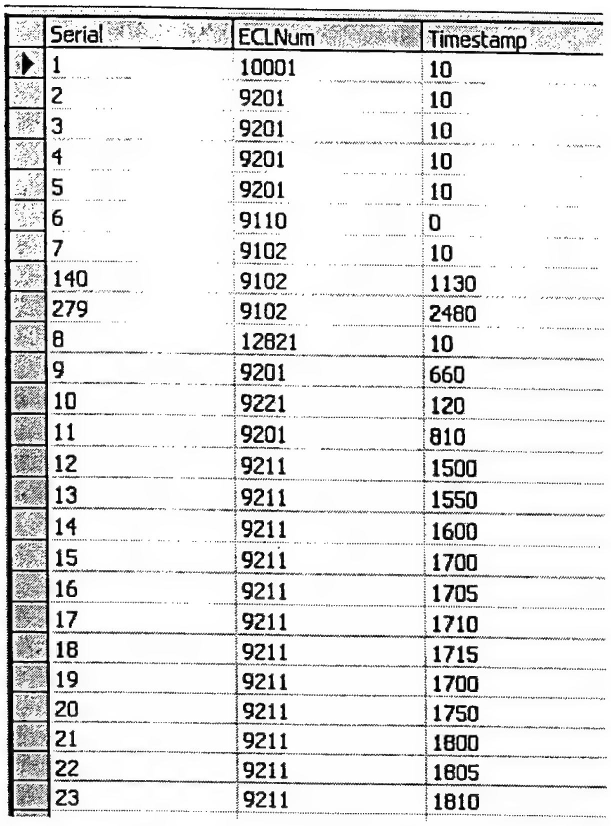
Quoted from "*Results of 1998 DC-ARM/ISFE Demonstrations Tests*" [Parker, et.al., 1999], page 46:

"Test arm1_07 was a simulated missile hit with warhead detonation test. The fire was simulated using two large Class A wood cribs in the forward corners of the Comm Center. The four large vents in the second deck, which simulates the blast damage of the deck, were open. This allowed free communication between Comm Center and CIC and led to ignition of the false deck. A high-volume rupture assembly was located off the fire main in the Ops Office. The Safety Team opened the rupture as the RRT entered CIC to simulate a high flow rupture resulting in the loss of fire main pressure. Blast damage other than the deck opening (i.e. fragmentation holes, jammed accesses, blocked accesses) was not included in this test. The intent of this test was to evaluate the manning organization at an intermediate step between the nondetonation and detonation test."

The following is a list of the events that DC-SIM can duplicate. It is part of the list indicated on page 104 of "*Results of 1998 DC-ARM/ISFE Demonstrations Tests*" [Parker, et.al., 1999]. Compartment ID used by DC-SIM and other Ids are inserted into the list whenever necessary to facilitate the user to understand the enclosed simulation data.

Event Time (min)	**Events**
-2:00	**Fire ignited in compartment 139 (Comm Center)**
2	DCA announced the upper boundary in CSMC/Repair 8.
3	Smoke reported in the Combat System Office at 2-22-4-L
4	Desmoking system was called away to rig for installed ventilation
5	DCA announce forward and aft boundaries at FR15 and FR29
7	False deck in compartment 218 (CIC) ignited
8	**Investigator reported a class A fire in compartment 218 (CIC)**
9	**Fire main restoration. Rupture initiated in compartment 113 (Ops office). (A rupture is set on pipe with ID 85. Its connection in fire main is like: valve (ID85)-----Pipe (ID85)-----junction (ID86). Valve 85 is originally open, so closing this valve isolates the rupture.**
10	DCA reported loss of the fire main on both the port and starboard sides
13.5	**DCA asked DCRS 2 to manually close valve 2-23-1 (Valve with ID 85), which restored the starboard side fire main.**
25	Instrumentation showed that water was applied to the Class A fire in compartment 218 (CIC).
30	**Fire in compartment 139 (Comm Center) was reported out by the Scene Leader**
36	Desmoking using AMR No. 1.
44.5	**Fire in compartment 218 (CIC) was reported out. This may have been a delayed report.**
53	**Flooding in compartment 113 (Ops Office) was discovered and stopped. (In the simulation, the flooding was caused by the rupture on pipe 85 set at 9th minute, and was already isolated at 13.5 minute)**

DC-SIM duplicates this scenario by reading in commands in the following ECL table (created automatically by an independent scenario generator), and acting correspondingly:



Serial	ECLNum	Timestamp
1	10001	10
2	9201	10
3	9201	10
4	9201	10
5	9201	10
6	9110	0
7	9102	10
140	9102	1130
279	9102	2480
8	12821	10
9	9201	660
10	9221	120
11	9201	810
12	9211	1500
13	9211	1550
14	9211	1600
15	9211	1700
16	9211	1705
17	9211	1710
18	9211	1715
19	9211	1700
20	9211	1750
21	9211	1800
22	9211	1805
23	9211	1810

Command 10001 with serial 1 ignites compartment 139 at time 10 second. The compartment number is not captured in the screen shot due to space limitations. (The ECL table is very wide).

The four 9201 commands set 3 ruptures on the deck connecting Comm Center and CIC, and open a hatch on the same deck. The purpose is to simulate a rupture caused by detonation. Again, only part of the table is shown. Other parameters are specified in String1, Num1 fields in the far right end.

Command 9110 starts the simulation.

Command 9201 with serial 9 initiates a fire main rupture (at time 660 seconds) on pipe 85 with (unshown) parameter String1 = "FireMainRupture", and Num1 = 85.

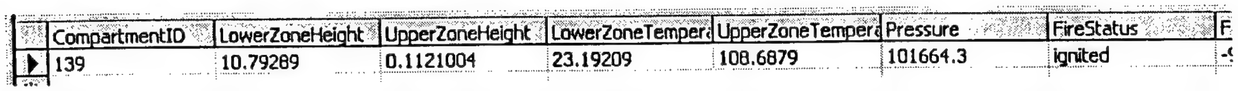
Command 9201 with serial 11 closes valve (ID85) at time 810. This is used to isolate the fire main rupture and thus restores fire main pressure and stops flooding in compartment 113 (Ops Office).

The rest of the 9211 commands are used to fight fires in compartment 139 and 218. Parameters of these fire-fighting commands are not captured due to space limits.

The bold part of the table is what we can show of how DC-SIM responds to the scenario. The other parts cannot be shown, like setting up fire boundaries, but DC-SIM does take that into account when simulating in real time.

The following is the output of DC-SIM, consisting of screen shots of the central database, with explanation whenever necessary.

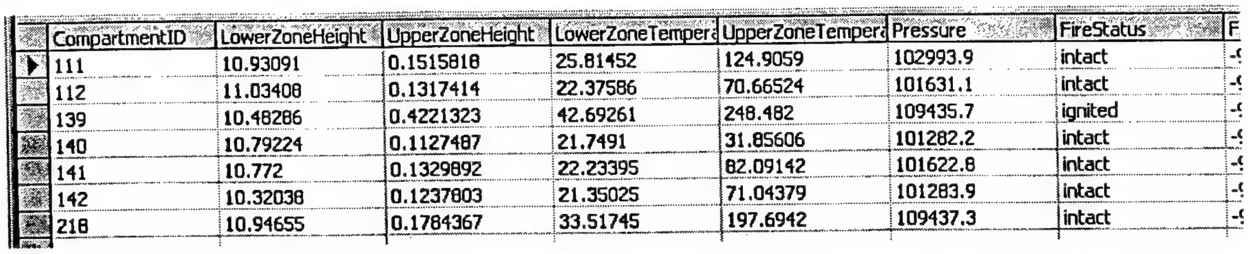
1. At time 25 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus	F
139	10.79289	0.1121004	23.19209	108.6879	101664.3	ignited	-

Compartment 139 (Comm Center) is ignited. The unit for zone heights is feet and temperature is Celsius degrees.

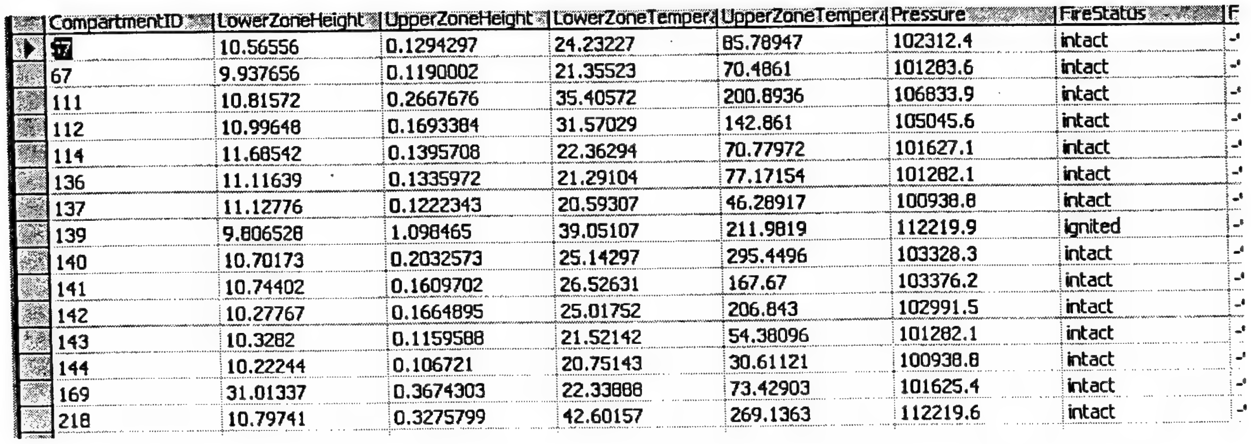
2. At time 265 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus	F
111	10.93091	0.1515818	25.81452	124.9059	102993.9	intact	-
112	11.03408	0.1317414	22.37586	70.66524	101631.1	intact	-
139	10.48286	0.4221323	42.69261	248.482	109435.7	ignited	-
140	10.79224	0.1127487	21.7491	31.85606	101282.2	intact	-
141	10.772	0.1329892	22.23395	82.09142	101622.8	intact	-
142	10.32038	0.1237803	21.35025	71.04379	101283.9	intact	-
218	10.94655	0.1784367	33.51745	197.6942	109437.3	intact	-

Temperature is increasing in the compartment centered at 139. Compartment 218's temperature is high due to vertical upward convection through open hatches.

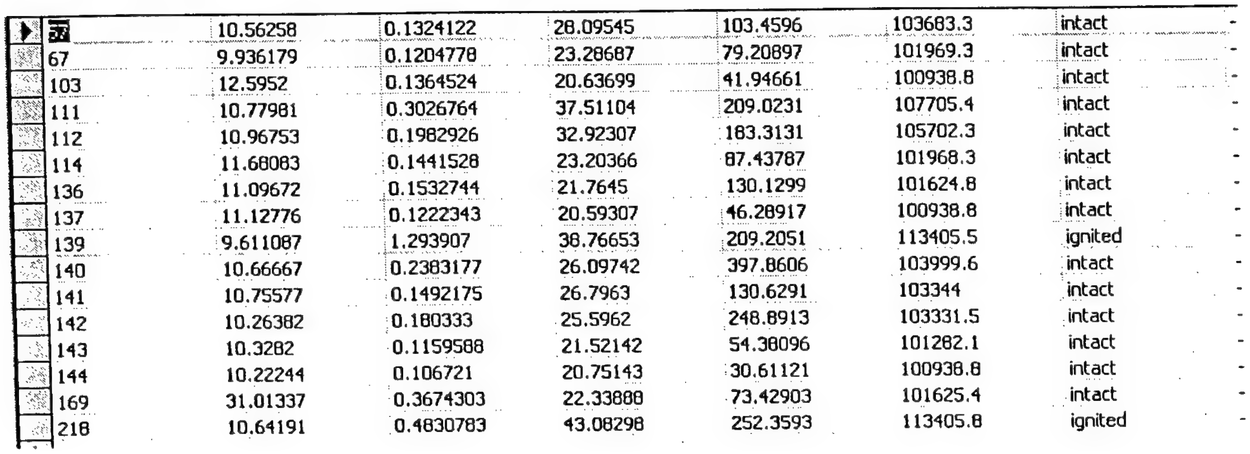
3. At time 625 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus	F
57	10.56556	0.1294297	24.23227	85.78947	102312.4	intact	-
67	9.937656	0.1190002	21.35523	70.4861	101283.6	intact	-
111	10.81572	0.2667676	35.40572	200.8936	106833.9	intact	-
112	10.99648	0.1693384	31.57029	142.861	105045.6	intact	-
114	11.68542	0.1395708	22.36294	70.77972	101627.1	intact	-
136	11.11639	0.1335972	21.29104	77.17154	101282.1	intact	-
137	11.12776	0.1222343	20.59307	46.28917	100938.8	intact	-
139	9.806528	1.098465	39.05107	211.9819	112219.9	ignited	-
140	10.70173	0.2032573	25.14297	295.4496	103328.3	intact	-
141	10.74402	0.1609702	26.52631	167.67	103376.2	intact	-
142	10.27767	0.1664895	25.01752	206.843	102991.5	intact	-
143	10.3282	0.1159588	21.52142	54.38096	101282.1	intact	-
144	10.22244	0.106721	20.75143	30.61121	100938.8	intact	-
169	31.01337	0.3674303	22.33888	73.42903	101625.4	intact	-
218	10.79741	0.3275799	42.60157	269.1363	112219.6	intact	-

Temperature is further increased. Some compartments have high temperature due to their location and shape.

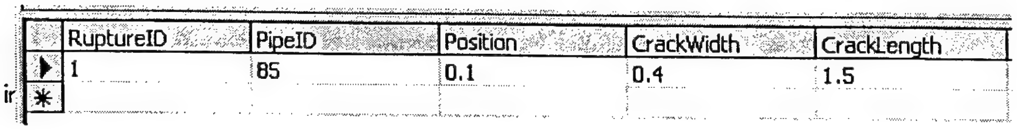
4. At time 700 seconds, part of CompartmentStatus table:



58	10.56258	0.1324122	28.09545	103.4596	103683.3	intact	-
67	9.936179	0.1204778	23.28687	79.20897	101969.3	intact	-
103	12.5952	0.1364524	20.63699	41.94661	100938.8	intact	-
111	10.77981	0.3026764	37.51104	209.0231	107705.4	intact	-
112	10.96753	0.1982926	32.92307	183.3131	105702.3	intact	-
114	11.68083	0.1441528	23.20366	87.43787	101968.3	intact	-
136	11.09672	0.1532744	21.7645	130.1299	101624.8	intact	-
137	11.12776	0.1222343	20.59307	46.28917	100938.8	intact	-
139	9.611087	1.293907	38.76653	209.2051	113405.5	ignited	-
140	10.66667	0.2383177	26.09742	397.8606	103999.6	intact	-
141	10.75577	0.1492175	26.7963	130.6291	103344	intact	-
142	10.26382	0.180333	25.5962	248.8913	103331.5	intact	-
143	10.3282	0.1159588	21.52142	54.38096	101282.1	intact	-
144	10.22244	0.106721	20.75143	30.61121	100938.8	intact	-
169	31.01337	0.3674303	22.33888	73.42903	101625.4	intact	-
218	10.64191	0.4830783	43.08298	252.3593	113405.8	ignited	-

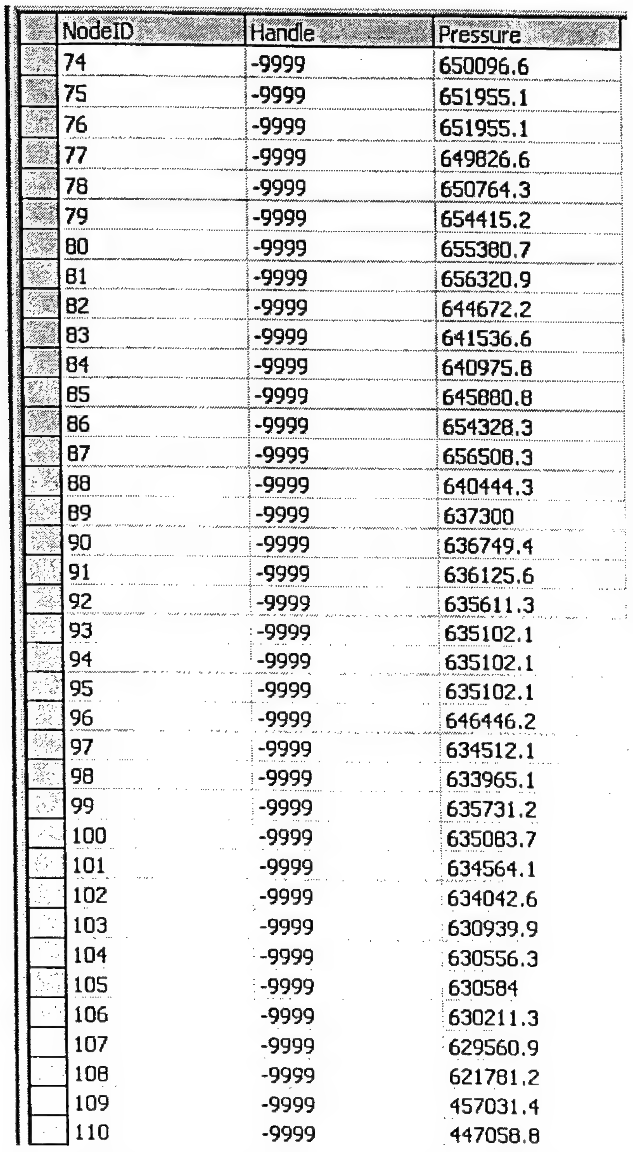
Fire has spread from compartment 139 (Comm Center) to Compartment 218 (CIC). Compartment 140 has high temperature because of its large contact area with compartment 139 and relatively small shape.

5. Fire main rupture specification (initiated at time 9 minutes). Part of the FiremainRuptures table:



RuptureID	PipeID	Position	CrackWidth	CrackLength
1	85	0.1	0.4	1.5

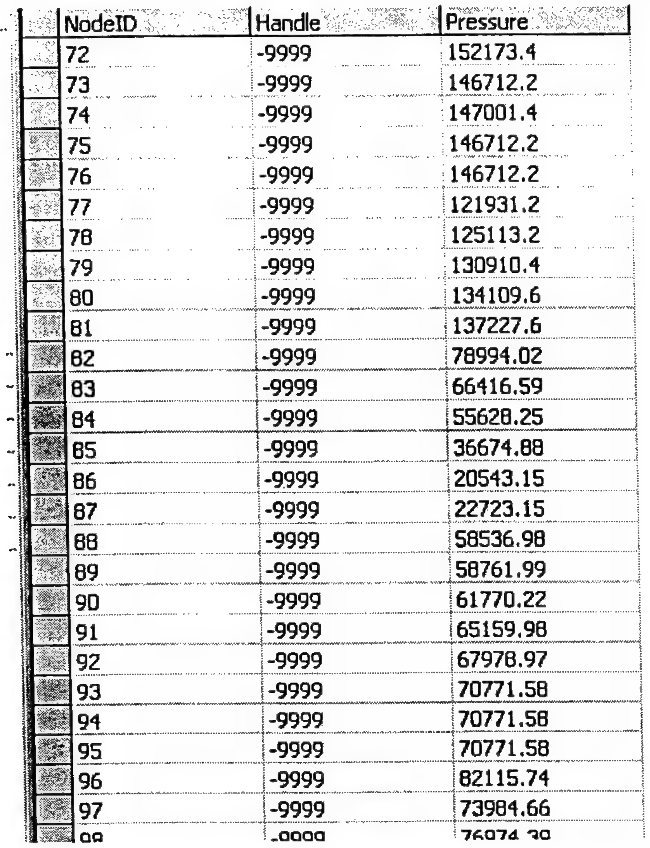
6. <u>Fire main pressure before the fire main rupture is set on pipe 85. Part of FiremainPressures table:</u>



NodeID	Handle	Pressure
74	-9999	650096.6
75	-9999	651955.1
76	-9999	651955.1
77	-9999	649826.6
78	-9999	650764.3
79	-9999	654415.2
80	-9999	655380.7
81	-9999	656320.9
82	-9999	644672.2
83	-9999	641536.6
84	-9999	640975.8
85	-9999	645880.8
86	-9999	654328.3
87	-9999	656508.3
88	-9999	640444.3
89	-9999	637300
90	-9999	636749.4
91	-9999	636125.6
92	-9999	635611.3
93	-9999	635102.1
94	-9999	635102.1
95	-9999	635102.1
96	-9999	646446.2
97	-9999	634512.1
98	-9999	633965.1
99	-9999	635731.2
100	-9999	635083.7
101	-9999	634564.1
102	-9999	634042.6
103	-9999	630939.9
104	-9999	630556.3
105	-9999	630584
106	-9999	630211.3
107	-9999	629560.9
108	-9999	621781.2
109	-9999	457031.4
110	-9999	447058.8

The unit of pressure is Pascal. Converted to PSI, its range is about 63~93 PSI.

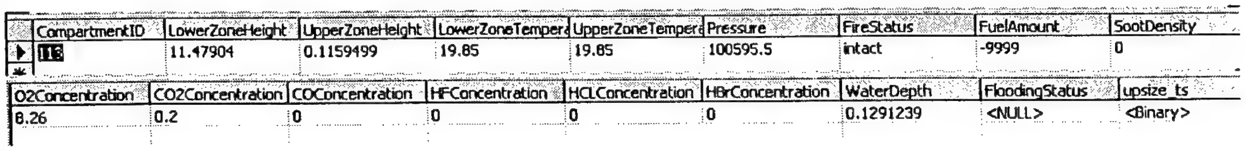
7. Fire main pressure after pipe rupture is set on pipe 85. Part of the FiremainPressures table:



NodeID	Handle	Pressure
72	-9999	152173.4
73	-9999	146712.2
74	-9999	147001.4
75	-9999	146712.2
76	-9999	146712.2
77	-9999	121931.2
78	-9999	125113.2
79	-9999	130910.4
80	-9999	134109.6
81	-9999	137227.6
82	-9999	78994.02
83	-9999	66416.59
84	-9999	55628.25
85	-9999	36674.88
86	-9999	20543.15
87	-9999	22723.15
88	-9999	58536.98
89	-9999	58761.99
90	-9999	61770.22
91	-9999	65159.98
92	-9999	67978.97
93	-9999	70771.58
94	-9999	70771.58
95	-9999	70771.58
96	-9999	82115.74
97	-9999	73984.66
98	-9999	76074.30

Converted to PSI, the pressure is about 3~22 PSI. The extent of pressure loss depends on the rupture's location and size.

8.Flooding caused by the fire main rupture leak. Part of CompartmentStatus table:

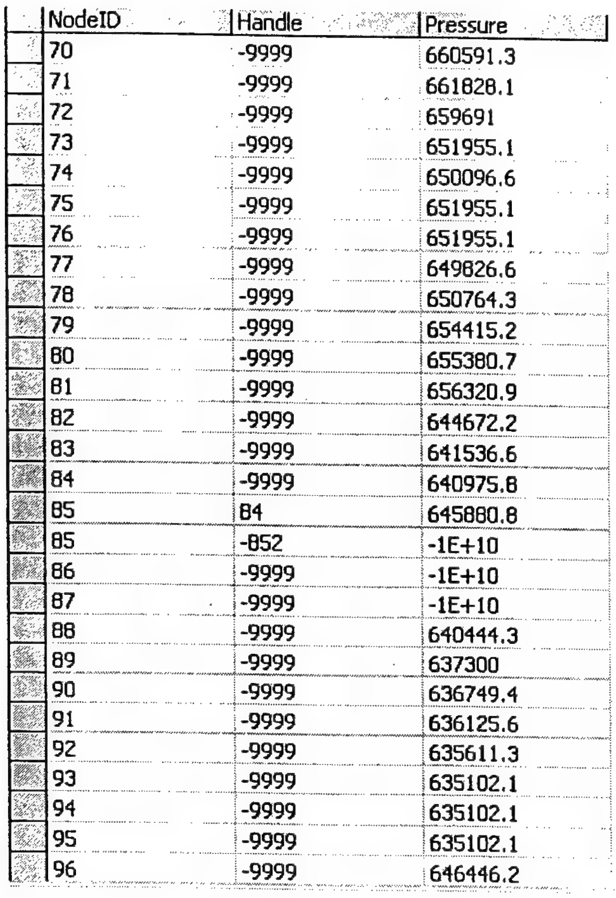


CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus	FuelAmount	SootDensity
113	11.47904	0.1159499	19.85	19.85	100595.5	intact	-9999	0

O2Concentration	CO2Concentration	COConcentration	HFConcentration	HCLConcentration	HBrConcentration	WaterDepth	FloodingStatus	upsize_ts
8.26	0.2	0	0	0	0	0.1291239	<NULL>	<Binary>

Flooding depth is in feet. At this time the depth is 0.129 foot in compartment 113 (Ops Office). This is the reading of flooding depth after step 8 (isolating fire main rupture).

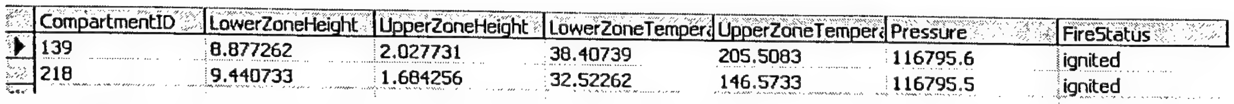
9.Fire main pressure after restoration (manually closing valve 2-23-1 to isolate rupture on pipe 85). Part of the FiremainPressures table:



NodeID	Handle	Pressure
70	-9999	660591.3
71	-9999	661828.1
72	-9999	659691
73	-9999	651955.1
74	-9999	650096.6
75	-9999	651955.1
76	-9999	651955.1
77	-9999	649826.6
78	-9999	650764.3
79	-9999	654415.2
80	-9999	655380.7
81	-9999	656320.9
82	-9999	644672.2
83	-9999	641536.6
84	-9999	640975.8
85	84	645880.8
85	-852	-1E+10
86	-9999	-1E+10
87	-9999	-1E+10
88	-9999	640444.3
89	-9999	637300
90	-9999	636749.4
91	-9999	636125.6
92	-9999	635611.3
93	-9999	635102.1
94	-9999	635102.1
95	-9999	635102.1
96	-9999	646446.2

It can be observed that pressure is restored, though part of the fire main (after valve 85) was isolated and thus had no pressure inside (notated by –1E+10, a negative infinity).

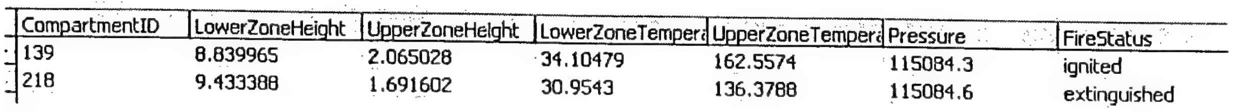
10. At time 1600 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus
139	8.877262	2.027731	38.40739	205.5083	116795.6	ignited
218	9.440733	1.684256	32.52262	146.5733	116795.5	ignited

Fire fighting in compartment 218 has been initiated. It can be observed that the temperature in compartment 218 (CIC) is decreasing.

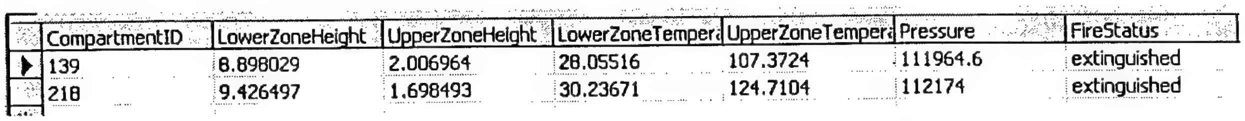
11. At time 1705 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus
139	8.839965	2.065028	34.10479	162.5574	115084.3	ignited
218	9.433388	1.691602	30.9543	136.3788	115084.6	extinguished

Continuous fire fighting in compartment 218 (CIC) put out the fire and the temperature decreased.

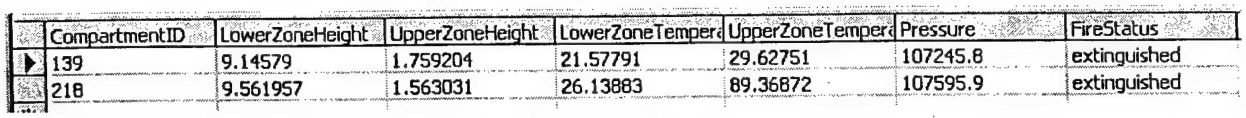
12. At time 1810 seconds, part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus
139	8.898029	2.006964	28.05516	107.3724	111964.6	extinguished
218	9.426497	1.698493	30.23671	124.7104	112174	extinguished

Fire fighting in compartment 139 (Comm Center) has also put out the fire and decreased the temperature.

13. At time 3000 seconds. part of CompartmentStatus table:



CompartmentID	LowerZoneHeight	UpperZoneHeight	LowerZoneTemper	UpperZoneTemper	Pressure	FireStatus
139	9.14579	1.759204	21.57791	29.62751	107245.8	extinguished
218	9.561957	1.563031	26.13883	89.36872	107595.9	extinguished

Temperature has decreased further due to heat dissipation.

Appendix C: List of Classes and Their Relationships

Despite the project's numerical nature, it is designed in an object oriented way to facilitate simulation of real time ship structure change. (The fire main simulator is an exception because a fire main solver is normally in centralized form). The project contains three types of objects: ship part objects, simulation control objects and I/O objects.

Each ship part object describes the physical properties of that object and its effect on neighboring objects' properties. For example, a door object not only contains members to determine its dimension, but also members to determine the airflow through it and thus affects its neighboring compartments' properties.

Each simulation control object controls simulation logic. They can be divided into main control objects (each for fire, flooding and fire main), and auxiliary control objects (parameter tuning).

I/O objects just provide a database interface and a human computer interface.

In the following we introduce in detail the classes defining these objects.

C.1 Ship Representation Classes

In this part we introduce ship classes representing ship structure. These classes form a complete map of a ship at any time.

C.1.1 The Container: CShipStructure

CShipStructure contains all the ship elements, such as compartments, bulkheads and doors, as well as methods used to initialize itself and to report ship status. This class doesn't include the fire main because it is not designed in a true object oriented way, and has little interaction with the rest of the system. (The fire main is a member of CShip, which also contains a CShipStructure member.)

CShipStructure is declared and implemented in ShipStructure.h and ShipStructure.cpp in subfolder "fire source".

C.1.2 Classes for Individual Parts Except Fire Main

Almost every member of CShipStructure corresponds to a ship element. They are instances of the classes discussed below.

C.1.2.1Ccompartment

This is the single most important class in the whole project. This class contains all the descriptions of the ship except what is provided by fire main. Specifically, it includes:

Dynamic information: air composition, temperature, pressure, remaining fuel amount and status, fire status, transmittance, water depth, flooding status, casualty control responses (fire fighting, de-flooding, de-smoking), and incoming/outgoing gas rate, smoke rate, heat rate and water flow rate.

Static characteristics: the ID's of the surrounding bulkheads (including decks and overheads) with the vents they contain (doors, hatches, scuttles, and ruptures), deck area, height, sensor information, and time constant (with its derivatives like temperature variation and heat dissipation delay).

Methods: combustion (plume effect), state updating based on incoming/outgoing rates and casualty control.

This class is declared and implemented in Compartment.h and Compartment.cpp in subfolder "fire source".

C.1.2.2CcompartmentContent

This is a simple class that contains descriptions of compartment contents. There can be up to three different types of contents in any compartment. The reaction properties of the fuels are described by members of class CShip that is the major control class and contains an instance of CShipStructure.

This class is declared and implemented in CompartmentContent.h and CompartmentContent.cpp.

C.1.2.3Cbulkhead

This class describes a bulkhead object. It contains: the ID of the bulkhead's two sides and the ID of the compartments that either side faces, the bulkhead's thickness, area, width and height, the heat flow through it, and method FindWallConduction to determine the heat flow. The heat flow is automatically collected by the two compartments in FindWallConduction.

This class is declared and implemented in Wall.h and Wall.cpp in subfolder "fire source".

C.1.2.4Cside

This class describes a side object. It contains the side's 4(or 3) vertices, its temperature, and the boundaries of its zones. It also contains information on whether or not a fire boundary has been set on it.

This class is declared and implemented in Side.h and Side.cpp in subfolder "fire source".

C.1.2.5Cvertex

This class just contains a vertex's coordinates. It is declared and implemented in Vertex.h and Vertex.cpp of subfolder "fire source".

C.1.2.6Cvent

This is the base class for a vent object. It contains basic descriptions like area, center height, ID of the bulkhead holding it and the vent's ID. It also contains dynamic information like various gas propagation rates and water flow rates. Methods used to determine air convection and water flow are also included, as well as several virtual functions for geometrical calculation that are used in calculating fluid dynamics. This class is declared and implemented in Vent.h and Vent.cpp in subfolder "fire source".

C.1.2.7Cdoor

Derived from CVent, this class describes a door object. It is relatively simple because it inherits all the complicated members of CVent, and only overrides the virtual functions declared in CVent. This class is declared and implemented in Door.h and Door.cpp in subfolder "fire source".

CHatch: Same as CDoor.

CScuttle: Same as CDoor.

CRupture: Same as CDoor.

C.1.3 Classes for Fire Main Parts

Fire main parts are not contained in class CShipStructure. Instead, they are members of class CFireMain . The following are classes describing parts of the fire main.

C.1.3.1Cposition

This class describes a point on the fire main by its x,y,z coordinates. The coordinates used here are the same as those used in describing class CVertex. (Actually these two classes are duplicated due to the fact that the fire main simulator and fire simulator were developed independently.) This class is declared in Position.h of subfolder "fire main source".

C.1.3.2CFMNode

This class describes a fire main node. A fire main node is a specific point in the fire main pipe network. It can be an elbow, an open valve, a side of a closed valve, a T-connection, a fireplug, or the intersection of a pipe with a bulkhead. This class contains a member on the node's position (a CPosition instance), whether or not it is a (closed) valve, its pressure, and graph characteristics like its father and sons in a spanning tree of the whole fire main system.

This class is declared and implemented in FMNode.h and FMNode.cpp of subfolder "fire main source".

C.1.3.3CFMEdge

This class describes a fire main edge. A fire main edge can either be a pipe segment or a pump. The class contains members on its hydraulic status like Reynolds number, flow rate, and friction coefficient, as well as methods used in calculating these variables.

This class is declared and implemented in FMEdge.h and FMEdge.cpp of subfolder "fire main source".

C.1.3.4Cpump

This class describes a fire main pump object. It contains the ID of the edge that the pump represents, the ID of the pump in the central database, and the pump's hydraulic characteristics. It also contains the method to determine the pump's flow rate based on its head gain, or vice versa.

This class is declared and implemented in Pump.h and Pump.cpp of subfolder "fire main source".

C.1.3.5CFMFirePlug

This is a class with member variables indicating the plug's ID and its status. See "fire main source\dbfm.cpp".

C.1.3.6CFMValve

Similar to CFMFirePlug.

C.2 Simulation Control Classes

These classes are not related to any ship object. Instead, they control the simulation process by either embodying numerical methods or assisting simulation threads. Besides the control classes, DC-SIM also uses a multimedia timer to gain real time performance. The time is also introduced in this section though it is not embodied in a class.

C.2.1 Classes for Simulation Threads

There are three simulation threads: fire/smoke, flooding and fire main. They are described by CFire, CFlooding and CFireMain respectively. Together with CShipStructure, they are held by class CShip which represents a complete simulated ship. Loading the ship, launching the threads and parameter tuning are implemented as members of class CMFShipDoc.

C.2.1.1Cfire

This class controls fire simulation with the timer. The main part of the fire simulation is implemented in CCompartment and CVent. So CFire just contains a pointer to the CShip object to gain access to all of the ship structure. CFire contains a fire parameter object for parameter tuning at run time. It also contains an event handle for synchronization between the fire thread and flooding thread to facilitate interaction between these two modules. CFire is declared and implemented in Fire.h and Fire.cpp of subfolder "fire source".

C.2.1.2Cflooding

Very similar to CFire. This class is declared and implemented in flooding.h and flooding.cpp in subfolder "flooding source".

C.2.1.3Cfire main

This class describes the fire main. Unlike fire and flooding simulation, it contains the fire main structure instead of putting it in CShipStructure. The fire main structure is stored in 4 arrays: pumps, edges, nodes and hoses, as well as an adjacent matrix representing the graph structure.

This class also contains methods for initializing and solving the fire main simulation, as well as on changing fire main structure at run time.

CFireMain is declared and implemented in CFireMain.h, CFireMain.cpp and dbfm.cpp of subfolder “fire main source”.

C.2.1.4Cship

This class represents a ship. It has a CShipStructure object holding ship structural information, a CFireMain object holding fire main information and solving fire main, a CFire object solving fire and smoke propagation, and a CFlooding object solving flooding spread. Besides these, it also contains a multimedia timer, 3 events for fire, flooding and fire main simulators to be activated when time is appropriate, a database object.

CShip doesn’t contain methods used to initialize the whole ship and launch the simulation. Instead, these methods are members of CshipDoc. The most important method CShip has is the ECL message retriever and processor. This method is called every time when the multimedia timer clicks.

This class is declared and implemented in CShip.h and CShip.cpp.

C.2.1.5CMFShipDoc

This class is part of the project’s MDI. It is not appropriate to place important simulation functions here. But it cannot be separated completely from the rest of the simulation project since it is the user interface for parameter tuning.

This class contains a parameter-tuning mechanism for each individual simulation thread. It also contains a ship loader that serves as the simulation launcher as well.

The class is declared and implemented in MFShipDoc.h and MFShipDoc.cpp.

Appendix D: Comparison of DC-SIM with CFAST on a Single Compartment Model

D.1 Abstract

To evaluate and improve the fidelity of heat conduction in the DC-SIM fire simulator, its predicted temperatures over time for a simple single-compartment fire were compared against those of the Consolidated Model of Fire Growth and Smoke Transport (CFAST). The boundary conduction parameter in DC-SIM was tuned to match the long-term temperature decay in a post-flashover compartment in the CFAST simulation.

D.1.1 The Single-Compartment Model

A cubical enclosed compartment of side length 3.33 m was constructed. The overhead, bulkhead and deck, all 0.3 m thick, were given thermal properties equivalent to the honeycombed steel bulkhead materials found in Navy vessels. This compartment deliberately lacked airflow connections to the external environment, in order to test fire spread and growth independent of airflow effects. However, heat conduction from the compartment to the environment could occur.

Ten seconds after simulation start, a fire in the compartment was set. Temperature, pressure, and smoke density data are recorded from both the upper and the lower zone of the compartment from t = 11 seconds up to and including t = 3600 seconds.

D.1.2 Setting the Fire

For an accurate comparison of DC-SIM and CFAST, the models' ignition conditions must match as closely as possible. Yet, their ignition processes differ. DC-SIM simply begins combustion of the available fuel at room temperature, using the amount of available fuel and oxygen, as well as the fuel's heat of combustion (a chemical constant), to calculate the fire's properties. In contrast, CFAST requires that the user specify any two of the following three parameters of the fire at one or more time points:

1. Heat of combustion (J/kg)
2. FMASS: The mass loss rate of the fuel – that is, the rate of fuel pyrolysis (kg/s)
3: FQDOT: The heat release rate of the fire (W)

Only two of these parameters are needed, since any one can be calculated from any other two. Heat of combustion is a constant, whereas FMASS and FQDOT can vary over time; thus, these latter two have more significance in starting and controlling a fire. In "real life," one starts a fire by adding heat energy (e.g. striking a match); by analogy, one may ignite a fire in CFAST by setting FQDOT to an appropriately high level at the ignition time. In this sense, CFAST's method more accurately reflects the ignition and growth of actual fires, upon which the heat energy and size of the ignition event have an impact (e.g. a newspaper ignited by a large gasoline explosion will be consumed faster than if ignited by a match).

Much effort has been expended in determining an appropriate initial FQDOT (heat release rate) to match the DC-SIM simulation most closely. In DC-SIM, the heat release rate is an internal variable, calculated at each timestep. By igniting a compartment and recording this rate over time, then using the resulting data as FQDOT inputs to CFAST, we attempted to conform CFAST's simulated fire to DC-SIM'S heat output, so that wall conduction parameters could be accurately compared.

D.2 Results

D.2.1 Problems with Matching Heat Release Rates in CFAST and DC-SIM

Unpredicted temperature results occurred when attempting to match heat release rates between DC-SIM and CFAST. It was found that the smaller the interval between the FQDOT time points specified, the more the maximum temperatures of both the upper and lower zones were constrained (see Figures D-1 and D-2). In a fire in an enclosed compartment of human-habitable size, one would expect that the temperature of the compartment at some time during the fire would increase to at least 600 K, the "flashover" point; yet, with an interval of 200s between specified FQDOT points, the peak temperature in the upper zone only slightly exceeded 500 K. For FQDOT data included at intervals smaller than 100s, the peaks were lowered even more.

It was also observed that adjusting the HOC (heat of combustion) parameter in CFAST (even to 16 times that in DC-SIM) had little if any effect on the peaks of the CFAST temperature curves. This suggests that the primary factor limiting heat production in CFAST (and presumably DC-SIM as well) was oxygen.

Figure D-1:

DC-SIM and CFAST lower zone temperatures over time,

with various FQDOT specifications in CFAST

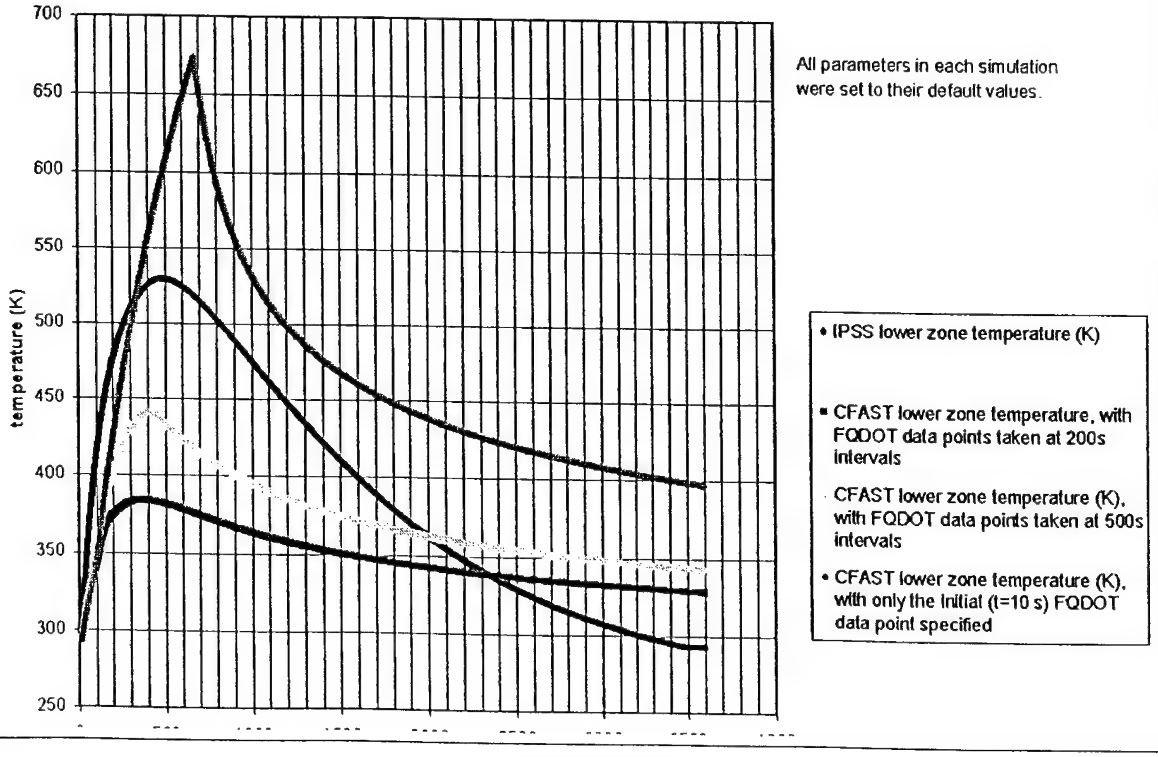
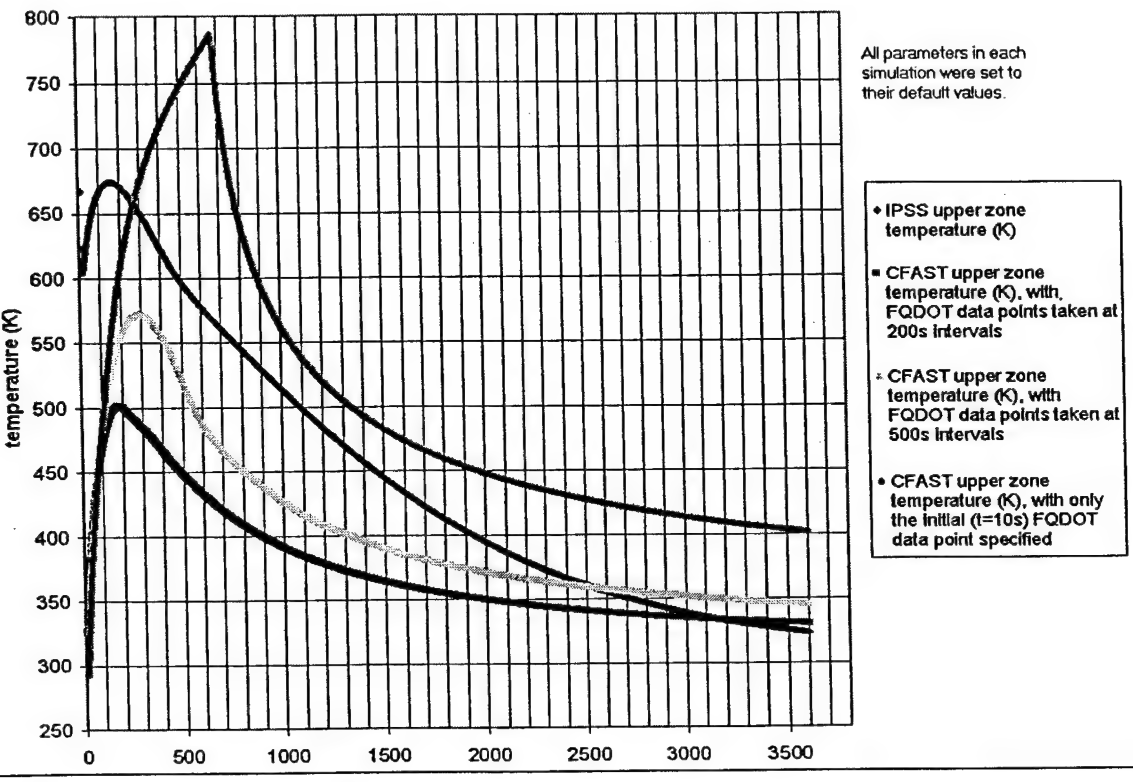


Figure D-2:

DC-SIM and CFAST upper zone temperatures over time, with various FQDOT specifications in CFAST



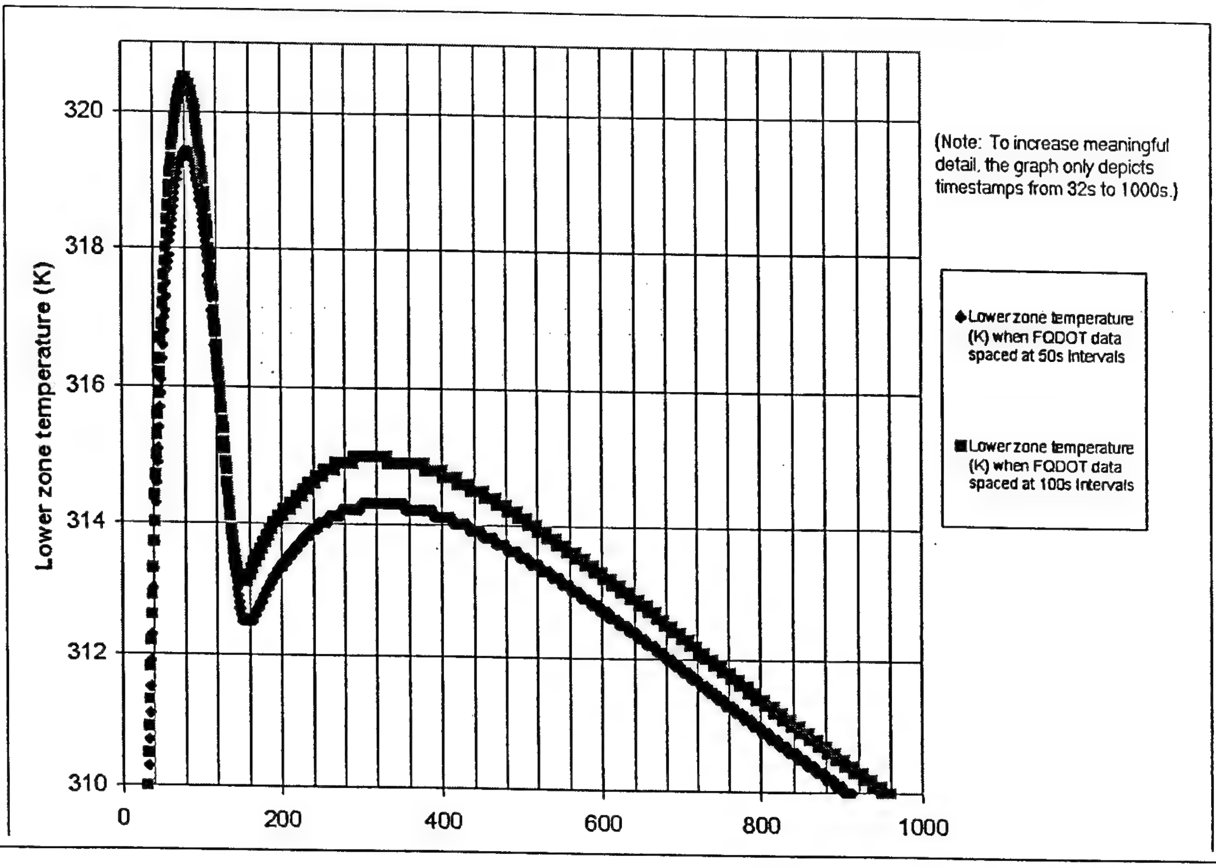
In addition, we found that numerical instability increased as the interval between FQDOT time points decreased. Although the CFAST simulation was set to run for 3600s, with FQDOT time points at 100s intervals, the system reported "division by zero" at 1909s; and with 50s intervals, "division by zero" occurred at 959s. (Note that the later stages of the fire are more important for measuring wall conductivity than the earlier stages, so that significant information is lost by premature termination of the algorithm.) The lower zone temperature curve exhibited an unexplained discontinuity with the FQDOT time point interval at 50s and 100s (see Figure D-3).

It was found that the qualitative shapes and peak temperatures of the DC-SIM and corresponding CFAST curves matched most closely when FQDOT data points were spaced at least 750s or more apart. This only held true when at least two FQDOT values were used – one at the start of the test, and one close to the end (within 600s of 3600s). For any interval width above 750s, up to and including 3000s (two data points, at 10s and 3010s), almost the same temperature results occurred as for the 750s interval test. Yet, upper and lower zone temperatures remained significantly lower than their respective maxima in the DC-SIM simulation (see Figure D-4).

Figure D-3:

CFAST lower zone temperatures over time,

with FQDOT data points spaced at 50s and at 100s intervals



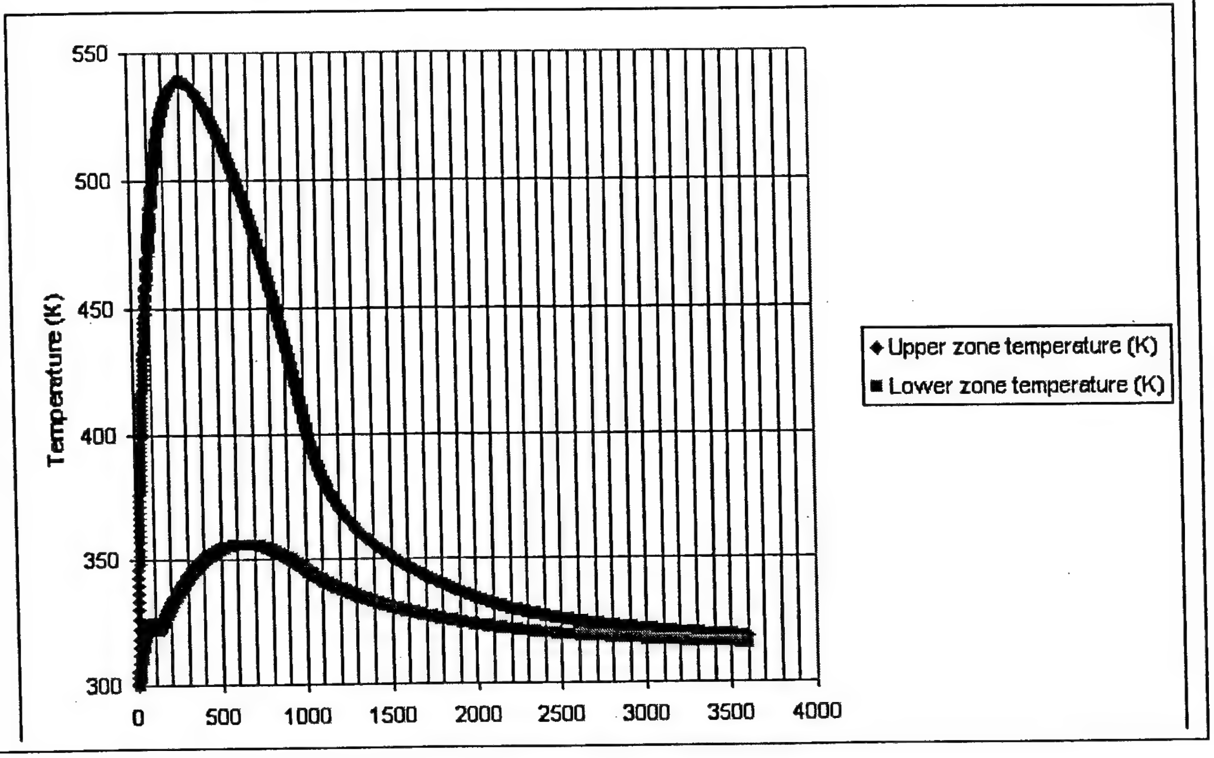
Experimentation also revealed that increasing CFAST's FQDOT parameters could elevate the temperature peak close to that of DC-SIM. However, such arbitrary manipulation of empirical data is unlikely to represent a fair comparison of the two simulations. Rather than forcing the temperature curves of the two simulations to match, we applied a procedure discussed below.

D.2.2 Boundary Conduction Comparison Methodology: Version 1

Since it is very difficult to control the parameters of both DC-SIM and CFAST so that only boundary conductivity is tested, we make the following assumption about fires in airtight compartments containing unlimited amounts of fuel: Once the temperature of the compartment begins to decrease after the peak, enough oxygen in the compartment has been consumed that the fire has stopped producing significant amounts of heat. Essentially, after a certain time after the peak, the compartment undergoes Newtonian cooling with the outside environment as a heat sink. In our model, the compartment is airtight, and thus the only means of heat transport to the outside world is via bulkhead conduction. Hence, the cooling rate should provide an approximate metric for the heat conductivity through the boundary.

Figure D-4:

CFAST upper and lower zone temperatures over time, with FQDOT data points spaced at 750s intervals



Newtonian cooling obeys the differential equation

(1) $y' = k(y - A)$

in which y is the current temperature, y' is the derivative of the temperature with respect to time, A is the ambient temperature (293 K), and k is the cooling rate (a negative real number). Its solution is

(2) $T(t) = A + C \cdot e^{kt}$

To justify the use of this equation as an approximation, suppose that we include a "driving function" $f(t)$ which describes the heat release rate of the fire in the "approximate Newtonian cooling" stage. For simplicity, assume in the following argument that the time $t = 0$ represents the beginning of this stage. Adding the driving function to the above differential equation results in the following:

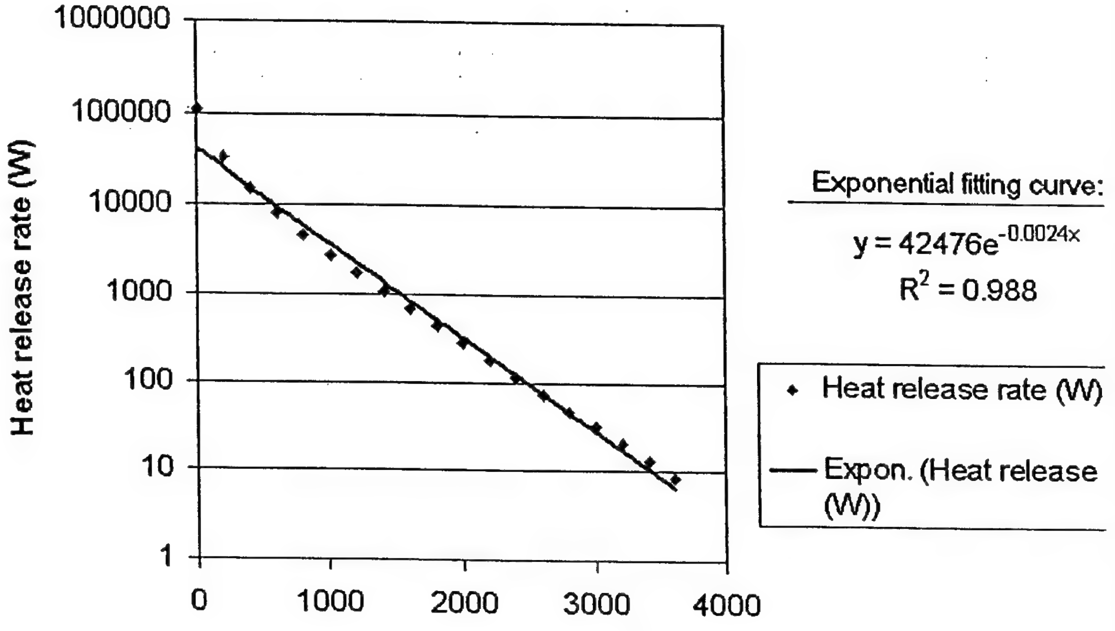
(3) $y' - ky = -kA + f(t)$

whose solution is:

(4) $T(t) = A + C \cdot e^{kt} + e^{kt} \int_0^t e^{-ks} f(s) ds$

for some positive constant C. Experience with fires in enclosed, oxygen-limited environments suggests that f will resemble exponential decay (since the fire's self-sustaining capacity is proportional to the amount of oxygen left in the compartment). Empirical values for $f(t)$ are given by the heat release rates that DC-SIM generates for our single-compartment model; from these data, we fit an exponential curve thereto (Figure D-5).

Figure D-5:
DC-SIM heat release rate (at default bulkhead conductivity of 0.2) over time: logarithmic vertical axis



Given this approximation for f with the empirical decay constant $R_d = -0.0024$, the integral term in the above expression becomes the following:

(5) $\int_0^t e^{(-k-0.0024)s} ds$

For a conservative k value of -0.0040, the term "-k-0.0024" evaluates to –0.0016, which, when compounded by the additional e^{kt} term outside the integral, causes the "non-Newtonian" part of Equation 5 to tend to 0. Hence, it seems reasonable that we may use Newtonian cooling as an approximation for the actual compartment cooling process.

D.2.3 Empirical Estimation of the Cooling Rate: Procedure 1

Assuming the Newtonian cooling approximation with cooling rate k, given two data points (t_0,y_0) and (t_1,y_1), we find that

(6) $k \approx \frac{\ln(y_1 - A) - \ln(y_0 - A)}{t_1 - t_0}$

We take data for cooling rates at $t_0 = 1500s$ and $t_1 = 2000s$, at which the compartment has been cooling for at least 500s. We only consider the lower zone cooling rate in this report, since the boundary layer effect can significantly alter the cooling rate near the overhead.

D.2.4 Results of Cooling Rate Comparison

In the three tables below are lower zone cooling rate values calculated using temperatures at 1500s and 2000s, over several simulation runs.

Table D-1

CFAST lower zone-cooling rates for various FQDOT data point intervals

FQDOT data point interval (in seconds)	Lower zone cooling rate
200s	-0.00032
500s	-0.000372
600s	-0.000375
750s	-0.000437
1000s	-0.000432
Data points at 10s and 3010s only	-0.000432

Table D-2

DC-SIM lower and upper zone cooling rates for various wall conductivity settings

DC-SIM wall conductivity setting	Lower zone cooling rate	Upper zone cooling rate
DC-SIM: default wall conductivity (0.2)	-0.00106	-0.0008
DC-SIM: 0.15 wall conductivity	-0.00078	-0.00062
DC-SIM: 0.1 wall conductivity	-0.00051	-0.00043

Table D-3

Comparison of CFAST cooling rates for doubled and unaltered FQDOT data

Simulation description	Lower zone cooling rate	Upper zone cooling rate
CFAST: regular FQDOT data, taken at 200s intervals	-0.00032	-0.00045
CFAST: doubled FQDOT data, taken at 200s intervals	-0.00034	-0.00043

A result, which supports our use of cooling rate as a comparator: in CFAST, doubling the values of all FQDOT data points had little effect on the cooling rate (as Table D-3 demonstrates). Thus, this rate does not depend upon the scale of fire output in our single-compartment model, permitting comparison between CFAST and DC-SIM without concern of equivalence of heat release rates.

The evidence of Table D-2 suggests that in DC-SIM, the cooling rate k seems to depend linearly on the boundary conductivity parameter. Linear regression predicts that k = -0.0055 (boundary conductivity), with 0.9999 correlation. For a cooling rate in DC-SIM equal to that in CFAST with unaltered FQDOT data taken at 750s intervals, the boundary conductivity must be 0.079. This is significantly less than the current default boundary conductivity of 0.2.

We ran a DC-SIM simulation of our single-compartment model, using 0.058 as the boundary conductivity constant (the predicted value when FQDOT data is given at 200s intervals). The resulting cooling rates: -0.00029 for the lower zone, and -0.0003 for the upper zone. For the lower zone, the error relative to the cooling rate in the equivalent CFAST simulation (using the above linear extrapolation) is only about 9.4 percent – indicating that our extrapolation above is reasonable. Also as predicted, the DC-SIM upper zone-cooling rate differs from the extrapolated prediction. We postulate that this difference is due to the boundary layer effect, whose parameter we will tune later.

D.2.5 Bulkhead Conduction Comparison Methodology: Version 2

For Version 1, we sampled temperature data at only two time points, 500s apart, to find the cooling rate. Curiosity led us to sample at every possible time, using backwards differences:

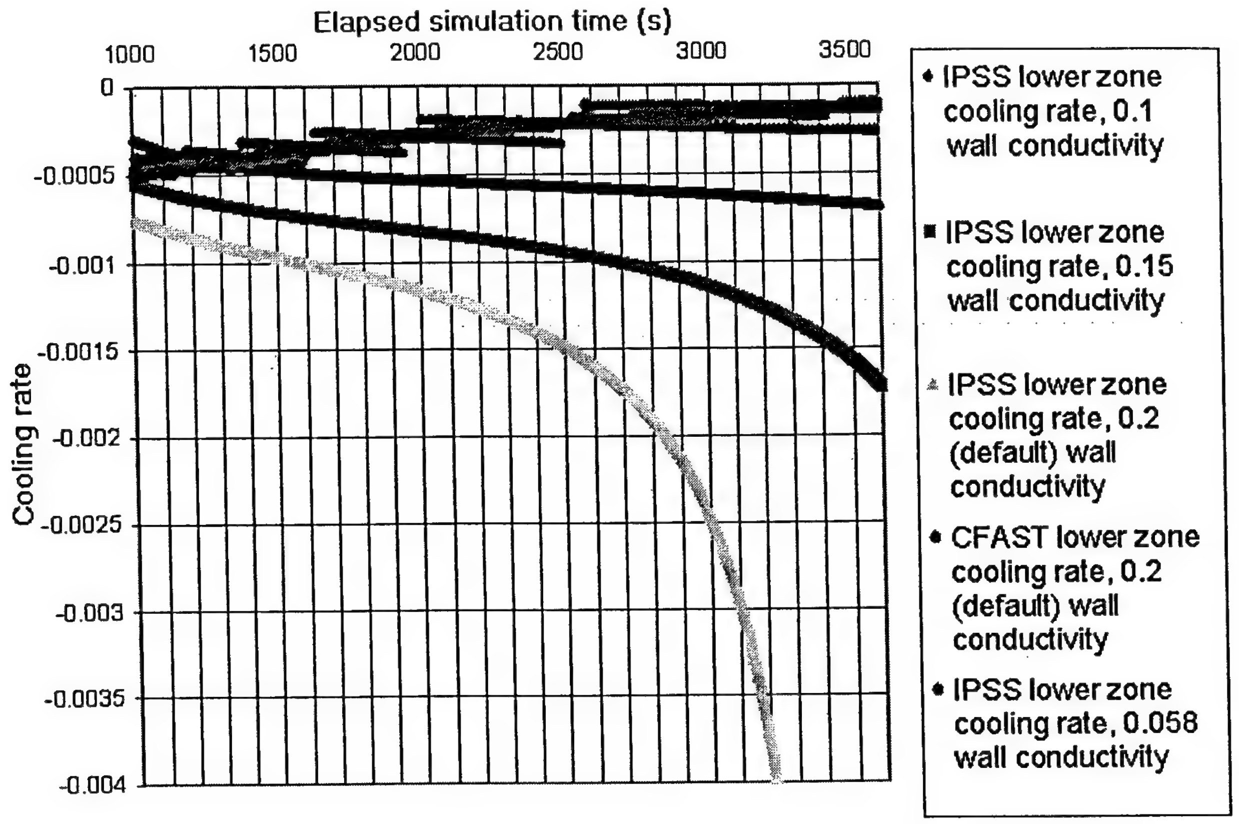
$$(7)\quad k = \frac{\ln(y_1 - A) - \ln(y_0 - A)}{t - (t-1)} = \ln(y_1 - A) - \ln(y_0 - A)$$

in which y_t is the current temperature, y_{t-1} is the temperature at the previous second, and t is the current simulation time. Naturally this cooling rate would only have meaning after the compartment reaches peak temperature and is well into approximate Newtonian cooling (at least after t=1000s). We found that the cooling rate in our fire simulations, unlike pure Newtonian cooling, changes slightly over time. In CFAST with 0.2 (default) bulkhead conductivity, it oscillates with a slowly increasing amplitude and a slowly decreasing frequency between zero and a small negative number. In DC-SIM with 0.058 wall conductivity, it follows a smooth curve, and is bracketed by the CFAST curve's oscillations until about t = 3000s, at which point it diverges downwards.

So we averaged the raw values over the past 20 to smooth out the oscillations. The CFAST (with FQDOT data taken every 200s) and DC-SIM lower zone-cooling rates matched very closely when the DC-SIM bulkhead conductivity was set to 0.058 (Figure 21). As an interesting aside, we observed significant oscillations in the DC-SIM cooling rate at 0.058 wall conductivity, although not at 0.1 or more wall conductivity.

Figure D-6:

DC-SIM and CFAST lower zone-cooling rates, with various DC-SIM wall conductivities



D.3 Evaluating Validity of Wall Conductivity Results

In order to compare DC-SIM with CFAST, it is paramount that both simulations be given the same parameters. In some cases, this could not be done.[2] Table D-4 contains a list of the relevant parameters of this type:

[2] In the case of FQDOT (heat release rate), the parameter was matched at a small finite number of discrete time points (separated by 200s time intervals). In the case of HOC, CFAST only accepts one heat of combustion value, while DC-SIM uses both a complete oxidation and an incomplete oxidation heat of combustion (two values) – implying that a direct assignment of one of the DC-SIM values to the equivalent CFAST parameter would be an approximating simplification.

Table D-4

Incomparable parameters between CFAST and DC-SIM

Parameter	**Explanation**
FQDOT (rate of heat release of the fire)	Needed to ignite compartment in CFAST; Only an internal parameter in DC-SIM.
HOC (heat of combustion)	DC-SIM uses two different HOC values: one for complete combustion and one for incomplete combustion. CFAST uses only one HOC value.
wall density	Necessary in CFAST; unused in DC-SIM.

For such parameters, we must demonstrate their independence from bulkhead conductivity. We have already shown that FQDOT (the heat release rate of the fire) does not affect the rate of cooling of the fire after flashover, and hence does not affect bulkhead conductivity. How do we know that the cooling rate for the lower zone is independent of the heat of combustion (HOC) parameter? The default HOC value in CFAST is $5.0*10^7$. We varied this parameter and measured the resulting cooling rates for the lower zone (cooling rate calculated using temperatures at 1500s and 2000s); they demonstrate that HOC does not correlate with the lower zone-cooling constant (see Table 5[3]).

Table D-5

Lower zone cooling constants for various

heat of combustion values in CFAST simulations

Heat of combustion in CFAST simulation	**Lower zone cooling rate**
$5.0*10^8$	-0.00033
$2.0*10^8$	-0.00033
$1.0*10^8$	-0.00032
$5.0*10^7$	-0.00034
$3.0*10^7$	-0.00033
$6.0*10^6$	-0.00033
$5.0*10^6$	-0.00032

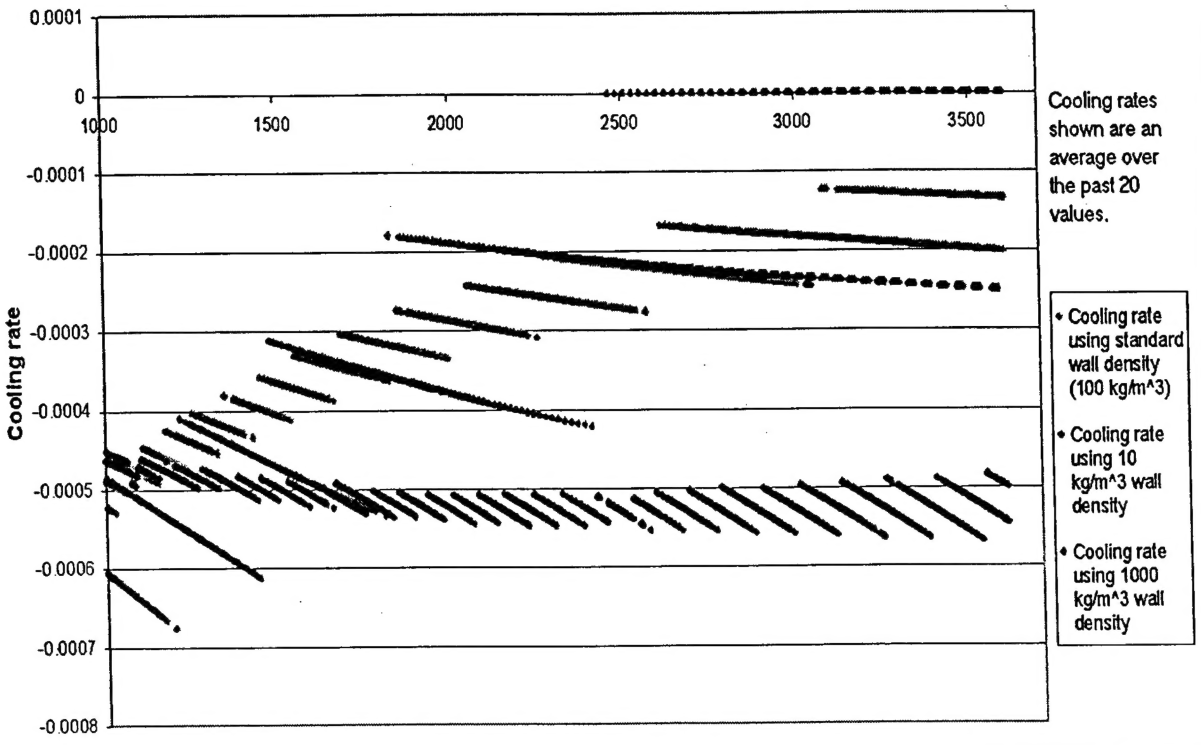
[3] It is interesting to note that the cooling constant values can vary in the 5th (least significant) decimal place, even between identical runs of the same scenario. Observe the cooling constant calculated for the default HOC in Table 5 and that calculated with the same (default) HOC in Table 3 – these were different runs of CFAST, but using the same parameters.

D.3.1 Effect of Bulkhead Material Density on Cooling Rate in CFAST Simulation

DC-SIM does not use bulkhead material density in its conduction calculations; all ship walls are assumed to be the same material, so tuning the bulkhead conduction parameter should tune the density as well. In contrast, CFAST requires that the density of the materials comprising the bulkheads of each compartment be specified, and factors density into its conduction calculations. Thus, in order that DC-SIM and CFAST represent the same fire conditions, the CFAST wall density parameter must be chosen correctly. As the graphs below indicate, the selection of this parameter critically affects both the value and the numerical stability of the cooling rate. Notice the unusual appearance of the cooling rate "curves" – rates oscillated significantly, and even after filtering through a moving average window, the unconnected appearance remains (Figure D-7).

Figure D-7:

Effects of wall density on cooling rate in CFAST



As the density decreases, the (post-flashover) cooling rate becomes more nearly constant – representing pure Newtonian cooling (as expected, since if the bulkheads have no density, then they have no mass, and thus can absorb no heat, and function merely to transfer heat between the compartment and the ambient environment). We attempted to reduce the bulkhead density to a very small value in CFAST to minimize its effect, but reducing the density below 2.5 kg/m^3 resulted in significant numerical instability, and results could not be computed for densities less than 1, due to (numerical) "matrix singularities" (as CFAST reported). (This is not a reason for complaint, since structural materials typically have a much higher density than 2.5 kg/m^3.) In

CFAST, the partial derivative of temperature with respect to time (for heat conduction) is proportional to the product of the Laplacian of the temperature and the inverse of the density, resulting in instability for small densities.

After arbitrarily picking a default wall density of 100 kg/m^3, we determined that this is a reasonable approximation of the actual density of the walls. The usual compartment wall material is honeycombed steel. While the density of solid steel is between 6000 and 8000 kg/m^3, the honeycomb structure results in numerous air spaces, such that only about 1/60 of the wall material is steel, resulting in a net wall density of about 100 kg/m^3.

As an interesting aside, the amplitude of the oscillation of the cooling rate – even considering that the cooling rates shown are averages over the previous 20 values – increases proportionally to the density.

D.4 Other Factors Affecting Results

D.4.1 Bulkhead Conductivity in CFAST

The CFAST material properties database also requires a conductivity coefficient for the bulkheads. We determined, however, that this coefficient has little correlation with the cooling rate of the compartment – though it correlates significantly with the temperature of the lower zone at t=3600s. We have no explanation for this phenomenon, and plan to research it further to determine its effect.

Table D-6

CFAST lower zone-cooling rates, given various CFAST bulkhead conductivities

Wall conductivity in the CFAST simulation	Resulting lower zone ending temperature (at t=3600s)	Resulting lower zone cooling rate
0.058	420.6	-0.00033
0.4	342.6	-0.00033
0.2	363.8	-0.00034

D.4.2 First Ten Seconds of the Simulation

Our specification that the fire begins at t=10s causes a minor problem of accuracy: when given FQDOT time points, CFAST interpolates between them to estimate the heat output of the fire. CFAST automatically assumes that the first FQDOT point is 0 W, at t=0s. So the temperature of the compartment in CFAST actually increases before the fire is supposed to ignite (at t=10s). However, this increase is minimal, a matter of 30-40K in the upper zone, 20K in the lower zone – and as will be discussed below, CFAST's temperatures throughout most of the simulation run are significantly lower than those of DC-SIM, implying that the initial temperature boost has little effect. Additionally, to measure the cooling rate, temperature values at t = 1500s and t =

2000s are collected – a sufficiently large time after start to smooth over any large differences caused by the 10s start time.

D.4.3 "Ceiling Jet" Effect

The "ceiling jet effect" provides an additional experimental parameter for CFAST simulations. A ceiling jet is a rapidly rising column of air above a fire in an enclosed compartment (with a ceiling). While numerical simulation of a ceiling jet adds significant computational cost, it more accurately represents the physical processes taking place in a compartment fire. Both DC-SIM and CFAST incorporate ceiling jet calculations into their models, although CFAST gives the user the option of turning off the ceiling jet calculations. We have performed CFAST simulations with and without this option, although for direct evaluation of DC-SIM performance, we compare it only against CFAST simulations with the ceiling jet activated.

D.4.4 Limitations of CFAST's Heat Conduction Model

CFAST's model of horizontal heat conduction transfers heat from a compartment through its bulkheads to a heat sink of ambient temperature. Heat is not conducted between horizontally adjacent compartments (although it is conducted between vertically adjacent compartments) (ibid. pp. 3, 4). In contrast, DC-SIM does account for heat conduction between compartments, both in the horizontal and vertical directions. While CFAST cannot be used to test DC-SIM'S horizontal heat conduction from one compartment to another, the general phenomenon of heat conduction can be measured in DC-SIM by examining heat transfer between a single compartment and the outside world through the walls. Consequently, focusing our attention on single-compartment models can make the two simulations comparable.

D.5 Conclusions

We recommend for now that the bulkhead conductivity parameter in DC-SIM be set to 0.058, on evidence of the lower zone cooling rate. Adjusting the boundary value coefficient may require further tweaking of the bulkhead conductivity parameter. The relationship between temperature at t=3600s, cooling rate, and the CFAST wall conductivity parameter requires further investigation.

D.6 Appendix to DC-SIM/CFAST Comparison

D.6.1 Heats of Combustion for Both Simulators

CFAST uses only one heat of combustion (HOC) parameter, whose default value is $5*10^7$. In contrast, DC-SIM uses two HOC values: one for complete combustion, whose default is $6*10^6$, and one for incomplete combustion, whose default is $3*10^6$.

Table D-7

Wall material properties required for a CFAST simulation

Wall material property name	Standard CFAST value	Corresponding DC-SIM parameter	Default value of DC-SIM parameter
conductivity (W/m/K)	0.2	conductivity	0.2
specific heat (J/kg/K)	559	---	---
density (kg/m^3)	100	---	---
thickness (m)	0.3	thickness	0.3
emissivity (dimensionless)	0.9	?	?

D.6.2 Version Information

CFAST version 3.16 was used for all of the above tests. The DC-SIM version employed in all the above tests stems from the 2 March 2000 release (with modifications for data collection completed on 24 March 2000).